DN. 0117-...
ATTY. K. OKADA
Date: 4-22-04
BSKB 703-205-8000

7-22-4

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

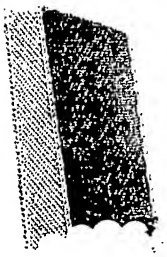
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　４月２４日

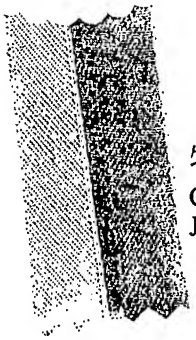
出願番号
Application Number: 特願２００３－１２０６００

[ST. 10/C]: [ＪＰ２００３－１２０６００]

出願人
Applicant(s): シャープ株式会社



CERTIFIED COPY OF
PRIORITY DOCUMENT



２００３年１２月１６日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫

出証番号　出証特２００３－３１０４４１３

【書類名】　特許願
【整理番号】　03J00997
【提出日】　平成15年 4月24日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F 17/50
【発明者】
　【住所又は居所】　大阪府大阪市阿倍野区長池町２２番２２号　シャープ株式会社内
　【氏名】　岡田　和久
【特許出願人】
　【識別番号】　000005049
　【氏名又は名称】　シャープ株式会社
【代理人】
　【識別番号】　100078282
　【弁理士】
　【氏名又は名称】　山本　秀策
【選任した代理人】
　【識別番号】　100062409
　【弁理士】
　【氏名又は名称】　安村　高明
【選任した代理人】
　【識別番号】　100107489
　【弁理士】
　【氏名又は名称】　大塩　竹志
【手数料の表示】
　【予納台帳番号】　001878
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　0208587

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　動作合成システム、動作合成方法、制御プログラム、可読記録媒体、論理回路の製造方法および論理回路

【特許請求の範囲】

【請求項１】　ループ処理および非ループ処理を含む動作記述から、各種処理手段を示す各節点とデータの流れを示す入出力枝によりコントロールデータフローグラフを生成するコントロールデータフローグラフ生成手段と、該コントロールデータフローグラフを用いてレジスタトランスファレベルのハードウェア構成を自動合成する手段とを有した動作合成システムにおいて、

該コントロールデータフローグラフ生成手段は、該ループ処理に含まれる各節点を各パイプライン化ステージ毎に分けて複数回のループ処理の１回毎に該各パイプライン化ステージの処理を並行して実行することを示すコントロールデータフローグラフのループ処理部内に、該ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる各制御信号をそれぞれステージ毎の各節点にそれぞれ出力可能とするループ制御部が設けられた動作合成システム。

【請求項２】　前記ループ制御部は、前記各制御信号をそれぞれ前記ステージ毎の各節点にそれぞれ出力する複数のポートと、該複数のポートのうち所定の一または複数のポートに接続されて終了判定を行う終了条件比較節点とを有する請求項１または２記載の動作合成システム。

【請求項３】　前記ループ処理部は、所定の入力初期値が入力され、これを１サイクル毎にインクリメントした値を順次出力するループ用ポートと、該ループ用ポートからの出力値に対してループ処理の回数が規定回数に達したかどうかを判定する判定出力を前記ループ制御部に出力するループ回数判定節点と、前記ステージ毎の各節点とを有する相請求項１または２に記載の動作合成システム。

【請求項４】　前記ループ制御部は、前記ループ回数判定節点からの判定出力を受けて、該ループ制御部の各ポートから前記ステージ毎の各節点にそれぞれ、非ループ処理を行わせる各制御信号をそれぞれ出力する請求項３に記載の動作合成システム。

【請求項５】　前記ループ制御部の複数のポートは、前記パイプライン化ス

テージのステージ段数と同じビット数のシフトレジスタの機能を有している請求項２に記載の動作合成システム。

【請求項６】　前記複数のポートは、前記ステージ毎の各節点にそれぞれ、前記シフトレジスタ機能によって所定の入力初期値が１サイクル毎に順次シフトされた各制御信号をそれぞれ供給する請求項５に記載の動作合成システム。

【請求項７】　前記ステージ毎の各節点であって、前記ループ処理部の外部に対して作用する節点に対して、当該節点を前記ループ制御部からの制御信号により制御する請求項１に記載の動作合成システム。

【請求項８】　前記コントロールデータフローグラフ生成手段は、パイプライン化されていないループ処理のコントロールデータフローグラフに基づいて、パイプライン化されたループ処理のコントロールデータフローグラフを生成する請求項１に記載の動作合成システム。

【請求項９】　前記コントロールデータフローグラフ生成手段は、前記パイプライン化されていないループ処理のコントロールデータフローグラフに対して、ステージ境界部分とグラフの枝が交差する箇所に新たにポートを追加するポート追加手段と、並列処理を示すために各ステージを横方向に配置する横方向配置手段と、横方向に配置されたループ処理部の各ステージに対して、繰り返し間データ転送枝を接続処理する枝接続手段と、該ループ処理部内に前記ループ制御部を設けるループ制御部追加手段とを有する請求項８に記載の動作合成システム。

【請求項１０】　前記枝接続手段は、ループ処理部外から与えられる初期値およびループ処理部内で演算された値の何れか一方を選択してループ処理部内の変数に代入するセレクタ節点を生成し、前記ループ制御部追加手段は、何れの値が代入されるかが前記ループ制御部の制御信号によって制御されるように、該ループ制御部とセレクタ節点とを接続処理する請求項９に記載の動作合成システム。

【請求項１１】　ループ処理および非ループ処理を含む動作記述から、演算処理手段を示す各節点とデータの流れを示す入出力枝によりコントロールデータフローグラフを生成するコントロールデータフローグラフ生成工程と、該コントロールデータフローグラフを用いてレジスタトランスファレベルのハードウェア

構成を自動合成する工程とを有する動作合成方法において、

該コントロールデータフローグラフ生成工程は、該ループ処理に含まれる各節点を各パイプライン化ステージ毎に分ける工程と、複数回のループ処理の１回毎に該各パイプライン化ステージの処理を並行して実行するループ処理部を生成する工程と、該ループ処理部内に、該ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる制御信号を該各節点に出力可能とするループ制御部を生成する工程とを有する動作合成方法。

【請求項１２】　請求項１１に記載の動作合成方法の各処理手順をコンピュータに実行させる制御プログラム。

【請求項１３】　請求項１２に記載の制御プログラムが記録されたコンピュータ読み取り可能な可読記録媒体。

【請求項１４】　請求項１に記載の動作合成システムにより自動合成された回路構成に基づいて論理回路を製造する論理回路の製造方法。

【請求項１５】　複数の論理演算手段による並列演算を繰り返し行うループ処理部内に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる各制御信号をそれぞれ複数の論理演算手段の少なくとも何れかに出力可能とするループ制御部が設けられた論理回路。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、例えば論理回路などの回路設計に用いられ、ループ処理を含む動作記述からパイプライン化されたコントロールデータフローグラフを生成する動作合成システム、これを用いてＲＴＬ（Register Transfer Level；レジスタトランスファレベル）の回路を自動合成する動作合成方法、その各処理手順をコンピュータに実行させる制御プログラム、これが記録されたコンピュータ読み取り可能な可読記録媒体、それらを用いた論理回路の製造方法および論理回路に関する。

【０００２】

【従来の技術】

従来、論理回路を含むシステムＬＳＩなどの大規模回路の設計において、動作記述からＲＴＬの論理回路を生成する動作合成処理が行われている。

【０００３】

この動作合成処理は、高位合成処理とも言われており、ハードウェアの構造に関する情報が含まれず、各処理のアルゴリズムのみが記述された動作記述から、ＲＴＬレベルのハードウェア（回路図）が自動的に合成される。

【０００４】

例えば、非特許文献１には、Ｃ言語をハードウェア設計用に拡張した言語を動作記述言語として、ハードウェア（回路図）を合成することが可能なコンピュータシステムである動作合成システムが開示されている。

【０００５】

また、非特許文献２には、従来の動作合成技術の詳細が記載されており、コントロールデータフローグラフ（ＣＤＦＧ）から所望のハードウェア（回路図）を得る処理の詳細について記載されている。

【０００６】

以下に、従来の動作合成技術を用いて、動作記述から所望の回路図を自動的に合成する動作合成装置による動作合成方法（高位合成方法）について簡単に説明する。

【０００７】

この動作合成装置は、コンピュータシステムにて構成され、例えば、後述する図示しないＣＤＦＧ生成手段と、スケジューリング処理手段と、アロケーション処理手段と、データパス生成手段と、コントローラ生成手段とを有しており、図１０の動作合成方法の各処理手順を順次行うことによりＲＴＬ（レジスタトランスファレベル）のハードウェア（回路図）を自動合成（自動設計）する。

【０００８】

図１０に示すように、まず、ステップＳ１のコントロールデータフローグラフ（ＣＤＦＧ）生成では、ＣＤＦＧ生成手段がアルゴリズム記述（動作記述）におけるデータの流れを解析し、ＣＤＦＧと称されるモデルを作成する。ＣＤＦＧは、演算を表す節点と、データの流れを表す枝とによって構成されている。各節点

は、入力枝と出力枝とによって接続されており、入力枝は演算に与えられるデータ、出力枝は演算結果のデータを表す。また、各節点はそれぞれ、演算の種類に関する情報も有している。

【０００９】

例えば図１１に示すＣ言語で記述された動作記述は、図１２に示すＣＤＦＧによって表される。

【００１０】

図１２のＣＤＦＧは、二つの乗算を表す節点１および２と、一つの加算を表す節点３とを含み、節点１で入力データａおよびｂが乗算された結果と、節点２で入力データｂおよびｃが乗算された結果とが、節点３で加算され、その結果が出力データＸとして出力されることを表している。

【００１１】

このＣＤＦＧは、コンピュータ（計算機）上では、例えば図１３に示すようなデータ構造で表される。

【００１２】

図１３のデータ構造では、節点はＮｏｄｅ構造体で表され、各節点固有の節点番号ｎｏｄｅ＿ｉｄを有している。また、ｉｎ＿ｅｄｇｅ配列には入力枝の枝番号が、ｏｕｔ＿ｅｄｇｅ配列には出力枝の枝番号が、複数記憶される。図１３の例に示される節点は２入力１出力の演算であり、ｉｎ＿ｅｄｇｅは二つの要素、ｏｕｔ＿ｅｄｇｅは一つの要素を有している。また、ｏｐ＿ｔｙｐｅには加算・減算・乗算などの各種演算の種類を表す番号が記憶されている。

【００１３】

また、枝はＥｄｇｅ構造体で表され、各枝固有の枝番号ｅｄｇｅ＿ｉｄを有している。ｆｒｏｍ＿ｎｏｄｅおよびｔｏ＿ｎｏｄｅには、その枝が接続された節点の節点番号が格納されている。

【００１４】

図１３のようなデータ構造により、コンピュータ（計算機）上のメモリ内にＣＤＦＧの各節点間の接続が記憶されている。

【００１５】

また、ＣＤＦＧ上で節点を接続する場合や、ある節点の入出力につながる別の節点を見つける場合には、以上のようにして計算機上のメモリ内に記憶されたデータの各要素に節点や枝の番号を登録し、またはそれを参照する。

【００１６】

以下では、説明を分かり易くするために、図１２のＣＤＦＧを視覚的に表し、節点を枝で接続したり、節点に枝で接続された別の節点を参照しながら説明を行う。

【００１７】

次に、図１０のステップＳ２およびステップＳ３の各処理では、ステップＳ１で生成されたＣＤＦＧに対して、スケジューリング処理手段がスケジューリング処理を行い、さらに、アロケーション処理手段がアロケーション処理を行う。

【００１８】

スケジューリング処理では、ＣＤＦＧ中の各節点をいつ実行させるかを決定する処理である。スケジューリング処理は、ＣＤＦＧ中の各節点がいくつかのステップに分けられる。一つのステップに含まれる節点は、クロックの変化の１サイクル中に実行される。

【００１９】

アロケーション処理はバインディング処理とも言われ、ＣＤＦＧ中の枝で表されるデータを格納するレジスタを決定する処理と、ＣＤＦＧ中の節点で表される演算を、どの演算器を用いて行うかを決定する処理とからなっている。動作合成方法（高位合成方法）によっては、このアロケーション処理がスケジューリング処理の前に行なわれる場合もある。

【００２０】

次に、図１０に示すように、ステップＳ４およびステップＳ５の各処理では、スケジューリング結果およびアロケーション結果を元に、データパス生成手段がデータパスを生成し、さらに、コントローラ生成手段がコントローラを生成することによりＲＴＬ（レジスタトランスファレベル）のハードウェア（回路図）が自動合成（自動設計）される。このように、ＣＤＦＧからハードウェア（回路図）を得る処理の詳細については上記非特許文献２に記載されており、また、その

一例が特許文献１に記載されている。

【００２１】

ところで、実際の設計回路では、動作記述中にループ処理が含まれていることが多い。このループ処理は、多数回繰り返して実行されるため、回路全体の処理時間のうち、多くは、ループ処理によって占められる。

【００２２】

したがって、回路の合成処理を高速化したい場合には、このループ処理を高速化することが有効である。例えば、画像処理や音声処理などのように、ある一定の時間内に処理を終える必要があり、高速性が要求される回路設計の場合には、このようなループ処理の高速化が必須である。

【００２３】

このようなループ処理の高速化という問題を解決する方法の一つとして、ループ処理をパイプライン化する方法が挙げられる。

【００２４】

従来のループ処理のパイプライン化方法としては、例えば非特許文献３に開示されているような方法が挙げられ、この方法は例えば特許文献２の図３（ｅ）に図示されている。

【００２５】

以下に、従来のループのパイプライン化方法について、図１４に示すようなループ処理を含む動作記述を一例として簡単に説明する。図１４では、ｉ＝０からｉを一つづつインクリメントしながらｉ≦１０の条件を満たす間、ｆ（ｉ）、ｇ（ｉ）およびｈ（ｉ）の演算を行うという各処理を表す動作記述である。

【００２６】

図１５は、図１４に示す動作記述について、ループ処理をパイプライン化しない場合のＣＤＦＧを示す図である。

【００２７】

図１５において、矩形１０はループ処理部を表し、ループ処理部１０内のＣＤＦＧが繰り返し実行される。ループ処理部１０の上端と下端に示す丸印１２および１３はループ処理部１０のポートを表し、図１５の変数ｉに対応する。ポート

１２に格納された変数ｉの値（データ）がインクリメント演算節点１４でインクリメントされて一つ増加するように変化し、ポート１３に至る。ポート１２と１３とは同じ変数を表しており、ポート１３に至ったデータは次の繰り返しで再びポート１２に戻されて使用されるようになっている。ループ処理部１０の外の節点１１は定数を出力する節点であり、まず変数ｉの初期値がポート１２に与えられる。

【００２８】

図１４の動作記述では、変数ｉのみがループ処理部１０内で繰り返し使用されるが、例えば２個以上の変数がループ処理部内で繰り返し使用される場合には、１２および１３のようなポートの対が、変数の個数分だけ設けられることになる。

【００２９】

節点１５では終了条件の比較が行なわれ、「真（ｉ≦１０）」であれば「１」、「偽」であれば「０」が出力される。

【００３０】

節点１６はループ処理部１０の終了を制御する特別な「ＥＸＩＴ」節点であり、終了条件比較節点１５からの入力データが「０」の場合（変数ｉが１０以下の場合）には、図示しないコントローラに対してループ１０の処理を終了し、次の状態に移行するように指示が行われる。

【００３１】

節点１７、１８および１９ではそれぞれ、図１４の動作記述中の関数ｆ、ｇおよびｈの各種演算処理が行われる。それぞれの節点１７、１８および１９には、変数ｉの値が入力されている。節点１７、１８および１９はそれぞれ、ステップ１、２および３にスケジューリングされている。

【００３２】

上記構成により、その処理動作を説明すると、クロックの１サイクル目で、１回目のループ処理のステップ１に含まれる節点１７で演算ｆ、即ちｆ（１）が実行され、２サイクル目で、１回目のループ処理のステップ２に含まれる節点１８で演算ｇ、即ちｇ（１）が実行され、３サイクル目で、１回目のループ処理のス

テップ３に含まれる節点１９で演算ｈ、即ちｈ（１）が実行される。

【００３３】

次に、４、５および６サイクル目で、２回目のループ処理のステップ１、２および３に含まれる節点１７、１８および１９でｆ（２）、ｇ（２）およびｈ（２）がそれぞれ順次実行される。したがって、ループ処理を１０回繰り返しすためには、３０サイクルの処理が必要となる。

【００３４】

なお、ＣＤＦＧでループ処理を表す方法には、決まった形式がなく、各文献毎に様々な表記がされているが、動作としては同一である。

【００３５】

次に、図１４の動作記述のループ処理をパイプライン化する場合について説明する。

【００３６】

ここでは、ループ処理のパイプライン化段数は３段とする。したがって、ループ処理を１回繰り返すときの処理を３ステージに分ける。なお、ステージとは、パイプライン動作をする場合の処理のまとまりのことである。

【００３７】

図１４の動作記述のループ処理は、パイプライン化されない場合には、１回の繰り返しが３ステップにスケジューリングされるだけであるので、３ステージに分けるためには、ループ処理の１ステップが１ステージ、２ステップが２ステージ、３ステップが３ステージに割り当てられる。

【００３８】

また、例えばパイプライン化されないループ処理が４ステップにスケジューリングされ、それを２ステージに分ける場合には、１および２ステップが１ステージに、３および４ステップが２ステージに割り当てられる。この場合、１ステージを実行するために２サイクルが必要となる。

【００３９】

パイプライン化された回路では、図１６に示すような各処理動作が行われる。

【００４０】

図１６に示すように、まず、クロックの１サイクル目ではｆ（１）のみが実行される。次に、２サイクル目では、ｇ（１）が実行されるのと同時に、２回目のループ処理の繰り返しが開始されてｆ（２）が実行される。さらに、３サイクル目では、ｈ（１）が実行されて１回目のループ処理の繰り返しが終了するのと同時に、２回目の繰り返しのｇ（２）と３回目の繰り返しのｆ（３）とが同時に実行される。以降、ｋサイクル目ではｈ（ｋ－２）、ｇ（ｋ－１）およびｆ（ｋ）が同時に実行される。さらに、１１サイクル目ではｈ（９）とｇ（１０）が、１２サイクル目ではｈ（１０）のみが実行されて、ループ処理が終了する。

【００４１】

このように、図１４の動作記述を実行するために、ループ処理がパイプライン化されない場合には３０サイクル必要であったが、ループ処理がパイプライン化された場合には１２サイクルでよい。

【００４２】

図１４の動作記述のループ処理を、従来の方法によってパイプライン化した場合のＣＤＦＧを図１７に示している。図１７のＣＤＦＧでは、スケジューリングも行われており、各節点が各ステップに分けられている。

【００４３】

図１７に示すように、ＣＤＦＧにおいて、ステップ１ではｆ（１）のみが実行され、ステップ２ではｇ（１）とｆ（２）とが同時に実行される。

【００４４】

次に、ステップ３に含まれるループ処理部２１の処理に進む。なお、このループ処理部２１では、１回の処理を繰り返すために１サイクルを必要とし、ループ処理の繰り返し回数に応じたサイクル数を必要とする。したがって、ステップ４の処理以降は、ステップ数とサイクル数とは異なる。

【００４５】

ループ処理部２１内には、１～３の各ステージの処理が並列に配置されている。節点２２は、ループ変数ｉの初期値３を与える節点である。節点２３および２４はそれぞれ、入力される値を１および２減算する節点である。したがって、１回目のループ処理では、節点２５、２６および２７によって、ｆ（３）、ｇ（２

）およびｈ（１）が同時に実行される。また、２回目のループ処理では、ｆ（４）、ｇ（３）およびｈ（２）が同時に実行され、３回目から７回目のループ処理が実行された後、８回目のループ処理では、ｆ（１０）、ｇ（９）およびｈ（８）が同時に実行される。節点２８で条件（１０以下）が比較され、その条件に一致した場合にループ処理部２１の処理が終了され、次のステップ４の処理へと進む。

【００４６】

さらに、ステップ４ではｇ（１０）およびｈ（９）が実行され、更に次のステップ５ではｈ（１０）のみが実行される。

【００４７】

図１７のＣＤＦＧ中、ループ処理部２１の本体に対して、その前に実行される部分２０の処理はプロローグ部と言われ、その後に実行される部分２９の処理部はエピローグ部と言われている。

【００４８】

このようなＣＤＦＧを作成することによって、ループ処理をパイプライン化して、より高速に処理することが可能となる。

【００４９】

【非特許文献１】

”ＡＣ－ｂａｓｅｄ Ｓｙｎｔｈｅｓｉｓ Ｓｙｓｔｅｍ，Ｂａｃｈ，ａｎｄ ｉｔｓ Ａｐｐｌｉｃａｔｉｏｎ” Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ ＡＳＰ－ＤＡＣ ２００１，２００１（ＩＥＥＥ Ｃａｔａｌｏｇ Ｎｕｍｂｅｒ：０１ＥＸ４５５，ＩＳＢＮ：０－７８０３－６６３３－６）

【非特許文献２】

”Ｈｉｇｈ Ｌｅｖｅｌ Ｓｙｎｔｈｅｓｉｓ，” Ｋｌｕｗｅｒ Ａｃａｄｅｍｉｃ Ｐｕｂｌｉｓｈｅｒｓ，１９９２（ＩＳＢＮ：０－７９２３－９１９４－２）

【非特許文献３】

”Ｐｅｒｃｏｌａｔｉｏｎ Ｂａｓｅｄ Ｓｙｎｔｈｅｓｉｓ” Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ Ｄｅｓｉｇｎ ＡｕｔｏｍａｔｉｏｎＣｏｎｆｅｒｅｎ

ｃｅ　１９９０，　ｐｐ．４４４－４４８　（ＩＥＥＥ）

【特許文献１】

特開２００１－２２９２１７号公報

【特許文献２】

特開２００１－１４２９３７号公報（図３（ｅ））

【００５０】

【発明が解決しようとする課題】

しかしながら、図１７に示すように、従来の方法でループ処理をパイプライン化した場合には、ＣＤＦＧ中のループ処理部の前後にそれぞれプロローグ部とエピローグ部とがそれぞれ必要とされ、ＣＤＦＧが複雑化する。このため、このＣＤＦＧからハードウェア（回路図）を動作合成すると、生成されるハードウェアの面積が大きくなり、チップの製造コストが高くなるという問題がある。

【００５１】

また、このような従来の方法でループ処理をパイプライン化したＣＤＦＧでは、プロローグ部とエピローグ部とは、ループ繰り返しの条件判断が行われずに必ず実行されてしまう。このため、プロローグ部の実行が終了する前にループ自体が終了してしまう場合には、正しい動作が行われなくなる。例えば、図１４の動作記述の場合、ループ処理の繰り返し回数を１回とするためには、ｆ（１）、ｇ（１）およびｈ（１）のみを実行する必要があるが、２サイクル目でｆ（２）が実行されてしまう。例えば、ｆ（２）の処理がメモリへの書き込みや外部との通信を行うという処理を含む場合には、メモリに誤った値が書き込まれたり、外部に不要な値が送出されたりするため、回路が誤動作するという問題がある。

【００５２】

このような誤動作を防ぐために、ループ処理の繰り返し回数が少ない場合を別に場合分けしたＣＤＦＧを導入することも考えられるが、この場合には、さらにＣＤＦＧが複雑化し、ハードウェア（回路図）面積が大きくなるという問題がある。

【００５３】

本発明は、上記従来の問題を解決するもので、従来のプロローグ部やエピロー

グ部を必要とせず、ハードウェアの面積増大とコスト増加を防ぐことができる動作合成システム、これを用いた動作合成方法、その各処理手順をコンピュータに実行させる制御プログラム、これが記録されたコンピュータ読み取り可能な可読記録媒体、それらを用いた論理回路の製造方法および論理回路を提供することを目的とする。

【００５４】

【課題を解決するための手段】

本発明の動作合成システムは、ループ処理および非ループ処理を含む動作記述から、各種処理手段を示す各節点とデータの流れを示す入出力枝によりコントロールデータフローグラフを生成するコントロールデータフローグラフ生成手段と、このコントロールデータフローグラフを用いてレジスタトランスファレベルのハードウェア構成を自動合成する手段とを有した動作合成システムにおいて、コントロールデータフローグラフ生成手段は、ループ処理に含まれる各節点を各パイプライン化ステージ毎に分けて複数回のループ処理の１回毎に各パイプライン化ステージの処理を並行して実行することを示すコントロールデータフローグラフのループ処理部内に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる各制御信号をそれぞれステージ毎の各節点にそれぞれ出力可能とするループ制御部が設けられたものであり、そのことにより上記目的が達成される。

【００５５】

また、好ましくは、本発明の動作合成システムにおけるループ制御部は、各制御信号をそれぞれステージ毎の各節点にそれぞれ出力する複数のポートと、複数のポートのうち所定の一または複数のポートに接続されて終了判定を行う終了条件比較節点とを有する。

【００５６】

さらに、好ましくは、本発明の動作合成システムにおけるループ処理部は、所定の入力初期値が入力され、これを１サイクル毎にインクリメントした値を順次出力するループ用ポートと、ループ用ポートからの出力値に対してループ処理の回数が規定回数に達したかどうかを判定する判定出力をループ制御部に出力する

ループ回数判定節点と、ステージ毎の各節点とを有する。

【００５７】

さらに、好ましくは、本発明の動作合成システムにおけるループ制御部は、ループ回数判定節点からの判定出力を受けて、ループ制御部の各ポートからステージ毎の各節点にそれぞれ、非ループ処理を行わせる各制御信号をそれぞれ出力する。

【００５８】

さらに、好ましくは、本発明の動作合成システムにおいて、ループ制御部の複数のポートは、パイプライン化ステージのステージ段数と同じビット数のシフトレジスタ機能を有している。

【００５９】

さらに、好ましくは、本発明の動作合成システムにおける複数のポートは、ステージ毎の各節点にそれぞれ、シフトレジスタ機能によって所定の入力初期値が１サイクル毎に順次シフトされた各制御信号をそれぞれ供給する。

【００６０】

さらに、好ましくは、本発明の動作合成システムにおいて、ステージ毎の各節点であって、ループ処理部の外部に対して作用する節点に対して、この節点をループ制御部からの制御信号により制御する。

【００６１】

さらに、好ましくは、本発明の動作合成システムにおけるコントロールデータフローグラフ生成手段は、パイプライン化されていないループ処理のコントロールデータフローグラフに基づいて、パイプライン化されたループ処理のコントロールデータフローグラフを生成する。

【００６２】

さらに、好ましくは、本発明の動作合成システムにおけるコントロールデータフローグラフ生成手段は、パイプライン化されていないループ処理のコントロールデータフローグラフに対して、ステージ境界部分とグラフの枝が交差する箇所に新たにポートを追加するポート追加手段と、並列処理を示すために各ステージを横方向に配置する横方向配置手段と、横方向に配置されたループ処理部の各ス

テージに対して、繰り返し間データ転送枝を接続処理する枝接続手段と、ループ処理部内にループ制御部を設けるループ制御部追加手段とを有する。

【００６３】

さらに、好ましくは、本発明の動作合成システムにおける枝接続手段は、ループ処理部外から与えられる初期値およびループ処理部内で演算された値の何れか一方を選択してループ処理部内の変数に代入するセレクタ節点を生成し、ループ制御部追加手段は、その何れの値が代入されるかがループ制御部の制御信号によって制御されるように、ループ制御部とセレクタ節点とを接続処理する。

【００６４】

本発明の動作合成方法は、ループ処理および非ループ処理を含む動作記述から、演算処理手段を示す各節点とデータの流れを示す入出力枝によりコントロールデータフローグラフを生成するコントロールデータフローグラフ生成工程と、このコントロールデータフローグラフを用いてレジスタトランスファレベルのハードウェア構成を自動合成する工程とを有する動作合成方法において、コントロールデータフローグラフ生成工程は、ループ処理に含まれる各節点を各パイプライン化ステージ毎に分ける工程と、複数回のループ処理の１回毎に各パイプライン化ステージの処理を並行して実行するループ処理部を生成する工程と、ループ処理部内に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる制御信号を該各節点に出力可能とするループ制御部を生成する工程とを有するものであり、そのことにより上記目的が達成される。

【００６５】

本発明の制御プログラムは、請求項１１に記載の動作合成方法の各処理手順をコンピュータに実行させるものであり、そのことにより上記目的が達成される。

【００６６】

本発明の可読記録媒体は、請求項１２に記載の制御プログラムが記録されたコンピュータ読み取り可能なものであり、そのことにより上記目的が達成される。

【００６７】

本発明の論理回路の製造方法は、請求項１に記載の動作合成システムにより自動合成された回路構成に基づいて論理回路を製造するものであり、そのことによ

り上記目的が達成される。

【００６８】

本発明の論理回路は、複数の論理演算手段による並列演算を繰り返し行うループ処理部内に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる各制御信号をそれぞれ複数の論理演算手段の少なくとも何れかに出力可能とするループ制御部が設けられており、そのことにより上記目的が達成される。

【００６９】

上記構成により、以下に、本発明の作用について説明する。

【００７０】

本発明にあっては、ループ処理および非ループ処理が含まれている動作記述から、パイプライン化されたＣＤＦＧを生成する際に、ループ処理に含まれる各節点を各パイプライン化ステージ毎に分けて複数回のループ処理の１回毎に各パイプライン化ステージの処理を並行して実行することを示すコントロールデータフローグラフのループ処理部内に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる制御信号を各節点に出力するループ制御部が設けられている。

【００７１】

一方、従来の構成では、ループ処理をパイプライン化する場合に、その前後のプロローグ部とエピローグ部とが条件判断無しに必ず実行されるため、プロローグ部の実行が終了する前にループ処理が終了すると、正しい動作が行われないおそれがある。また、プロローグ部とエピローグ部とがループ処理部とは別に設けられているため、ハードウェアの面積が大きくなる。

【００７２】

本発明では、プロローグ部とエピローグ部との機能をループ制御部としてループ処理部内に含めたため、従来のようにプロローグ部とエピローグ部とをループ処理部とは別に設ける必要がなく、誤動作やチップ面積増大を防ぐことができる。

【００７３】

また、単純な動作記述の場合には、パイプライン化されたＣＤＦＧを作成することは容易であるが、動作記述が複雑である場合には、パイプライン化されたＣＤＦＧを作成することは困難である。このような場合には、パイプライン化されていないループ処理のＣＤＦＧを変形してパイプライン化されたループ処理のＣＤＦＧを作成することが有効である。

【００７４】

ループ処理部内に含まれる節点のうち、ループ処理の外部に対して作用しない節点は、実行されても実行されなくても問題が生じないが、外部に対して作用する節点では、誤動作が生じるおそれがある。したがって、そのような外部に対して作用する節点では、ループ制御部からの出力制御信号によって制御する。

【００７５】

また、ループ処理部の外から与えられる初期値およびループ処理部内で演算された値のいずれか一方を選択してループ処理部内の変数に代入するセレクタ節点においても、いずれの値を代入するかをループ制御部の出力制御信号によって制御する。

【００７６】

さらに、ループ処理および非ループ処理の終了判定についても、ループ制御部からの出力制御信号によって制御することができる。

【００７７】

本発明によって設計された論理回路図に基づいて製造された論理回路は、ループ制御部が設けられているので、ループ処理部で誤動作が生じることなく、また、プロローグ部やエピローグ部が不要であるためチップ面積を縮小化することができる。

【００７８】

【発明の実施の形態】

以下、本発明の動作合成システムの実施形態について、図面を参照しながら説明する。

【００７９】

図１は、本発明の動作合成システムの実施形態における要部構成を示すブロッ

ク図である。

【００８０】

図１において、動作合成システム１００は、コンピュータシステムにて構成され、各部を制御して制御プログラムを実行する制御部１１１と、制御プログラムおよびそのデータなどが記録されたＲＯＭ１１２と、制御部１１１の制御時にワークメモリとして働くＲＡＭ１１３と、回路設計に必要な各種データが記録されたデータベース１１４と、ユーザによる操作指令入力を可能とする操作部１１５と、各種画像表示が為される表示部１１６とを有し、ループ処理および非ループ処理を含むアルゴリズムが記録された動作記述から、演算処理を示す節点とデータの流れを示す入出力枝により生成されたコントロールデータフローグラフを用いて、ＲＴＬ（レジスタトランスファレベル）の所望のハードウェア（回路図）を自動合成（自動設計）する。この場合、ＲＯＭ１１２には、ＦＤやＣＤ（光ディスク）などの可読記録媒体から制御プログラムおよびそのデータがインストールされている。

【００８１】

制御部１１１はＣＰＵ（中央演算処理装置）で構成されており、制御プログラムに基づいて、ＣＤＦＧ生成手段１１１Ａと、スケジューリング処理手段１１１Ｂと、アロケーション処理手段１１１Ｃと、データパス生成手段１１１Ｄと、コントローラ生成手段１１１Ｅとによる各機能を順次実行するようになっている。

【００８２】

ＣＤＦＧ生成手段１１１Ａは、アルゴリズム記述（動作記述）におけるデータの流れを解析し、演算などの各種処理機能を表す節点と、データの流れを表す枝とによって構成されるＣＤＦＧと称されるモデルを作成する。入力枝は演算などの各種処理機能に与えられるデータ、出力枝は演算結果のデータを表す。また、各節点はそれぞれ、演算の種類に関する情報も有している。また、ＣＤＦＧ生成手段１１１Ａは、ループ処理に含まれる各節点を各パイプライン化ステージに分け１回毎のループ処理に対して各パイプライン化ステージの各処理を並行して実行するようにしたＣＤＦＧのループ処理部内に、ループ処理が行われないときに非ループ処理を行わせる制御信号を各節点に出力可能とするループ制御部が設け

られる構成である。即ち、ループ処理部内にループ制御部が設けられており、ＣＤＦＧはループ処理部とループ制御部とを有する。

【００８３】

スケジューリング処理手段１１１Ｂは、ＣＤＦＧ中の各節点をいつ実行させるかを決定する処理である。スケジューリング処理手段１１１Ｂは、ＣＤＦＧ中の各節点をいくつかのステップに分ける。一つのステップに含まれる節点は、クロック変化の１サイクル中に実行される。

【００８４】

アロケーション処理手段１１１Ｃは、ＣＤＦＧ中の枝で表されるデータを格納するレジスタを決定するレジスタ決定手段と、ＣＤＦＧ中の節点で表される演算機能を、どの演算器を用いて行うかを決定する演算器決定手段とを有している。なお、動作合成システムによっては、このアロケーション処理手段１１１Ｃによる処理がスケジューリング処理手段１１１Ｂによる処理前に行われる場合もある。

【００８５】

データパス生成手段１１１Ｄはデータパスを生成する。

【００８６】

コントローラ生成手段１１１Ｅはコントローラを生成する。これによって、ＲＴＬ（レジスタトランスファレベル）のハードウェア（回路図）が自動合成（自動設計）される。

【００８７】

以下に、本発明のＣＤＦＧ生成手段１１１Ａにおけるループ処理のパイプライン化方法の各実施形態１，２について図面を参照しながら詳細に説明する。

【００８８】

（実施形態１）

本実施形態１では、図１４の動作記述に対して、本発明のループ処理のパイプライン化方法を適用した例について説明する。

【００８９】

図２は、図１４の動作記述に対して、本実施形態１のループのパイプライン化

方法によりループをパイプライン化したＣＤＦＧを示す図である。

【００９０】

図２のＣＤＦＧでは、図１７に示す従来の方法によるＣＤＦＧから、プロローグ部２０とエピローグ部２９とからなる非ループ処理が削除され、ループ処理を構成する各パイプライン化ステージに対して、その節点（演算処理手段）のループ処理と非ループ処理と制御する制御信号Ｖ１～Ｖ３を出力するループ制御部３１がループ処理部３０に設けられている。このループ制御部３１からの出力信号線（出力枝）はそれぞれ、ループ処理部３０を構成する各ステージ内で演算処理を行う節点に接続されている。また、繰り返し変数「ｉ」の初期値を与える節点４１の値は「１」に設定されている。

【００９１】

ここで、まず、ループ処理部３０におけるループ制御部３１について詳細に説明する。

【００９２】

図２において、ループ制御部３１には、パイプライン段数分の個数のループ制御変数が設けられている。この図２の例では、パイプライン段数が３段であるため、三つのループ制御変数Ｖ１、Ｖ２およびＶ３が設けられており、それぞれ、ポート３５、３６および３７に対応している。これらのループ制御変数Ｖ１、Ｖ２およびＶ３は全て、「０」または「１」を表す１ビットの変数である。

【００９３】

制御変数Ｖ１はループ処理をパイプライン化したときにステージ１に含まれる処理を制御するための制御信号として作用するものであり、制御変数Ｖ２はステージ２に含まれる処理を制御する制御信号として作用するものであって、制御変数Ｖ３はステージ３に含まれる処理を制御する制御信号として作用するものである。

【００９４】

制御変数Ｖ１には初期値「１」が節点３２から与えられ、それ以外のループ制御変数Ｖ２およびＶ３には初期値「０」が節点３３および３４からそれぞれ与えられる。また、ループ制御部３１では、制御変数Ｖ１およびＶ２の値が、次の繰

り返しではＶ２およびＶ３に転送されるように、ポート３５はポート３９へ、ポート３６はポート４０に接続されている。このループ制御部３１のＣＤＦＧは、１ビットのレジスタを三つ直列に接続した構成と同様の構成になっており、ハードウェア（回路図）としては３ビットのシフトレジスタによって実現される。また、ポート３８には、比較節点４３から終了条件の比較結果出力が入力されるようになっている。

【００９５】

また、クロックの各サイクルにおいて、Ｖ１、Ｖ２およびＶ３の値は、図２に示すように、Ｖ１が「１」のときにのみ節点４６でｆの演算処理が実行されるように、Ｖ２が「１」のときにのみ節点４７でｇの演算処理が実行されるように、また、Ｖ３が「１」のときにのみ節点４８でｈの演算処理が実行されるように、ポート３５、３６および３７と各節点４６、４７および４８がそれぞれ接続され、各節点４６、４７および４８が制御変数Ｖ１、Ｖ２およびＶ３によってそれぞれ駆動制御されるようになっている。これにより、演算ｆ、ｇおよびｈが、図５に示す回数と同じ回数だけ、行われるように制御することが可能となる。

【００９６】

次に、節点４９について説明を行う。

【００９７】

節点４９は、ポート３６および３７に接続され、制御変数Ｖ２およびＶ３が入力されて、Ｖ２が「０」、かつ、Ｖ３が「１」の場合に、ループ処理部３０の処理を終了するように図示しないコントローラに処理終了指示が与えられるようになっている。したがって、図３に示すサイクル１２で全処理が終了する。

【００９８】

節点４３において、ループ処理の回数が規定回数に達したか否かが判定されるのはサイクル１０であるが、その後も、図１７に示すようなエピローグ部２９に相当する回数分（２サイクル分）だけ、ループ処理部３０にて処理が続行されるようになっている。ループ処理としては１０回で終了している。

【００９９】

この方法によれば、ループの処理回数が固定値とされていない場合についても

、全く同様に、エピローグ部（非ループ処理）も含めた正しい実行回数だけ、ループ処理および非ループ処理を含む処理を実行することができる。

【０１００】

図２に示すＣＤＦＧにおいて、ループ変数ｉの初期値を与える節点４１の値は１に設定されている。これは、図１７に示すプロローグ部２０で実行されるｉ＝１、２のときの処理も、ループ処理部３０の本体内で行うためである。

【０１０１】

図１４の動作記述において、ループ処理は１０回繰り返されるが、Ｃ言語のｗｈｉｌｅループのように、ループを再度繰り返すか否かをループ内の条件式で判断する場合には、ループが１度だけ実行されて終了する場合も考えられる。例えば、図１に示すＣＤＦＧの節点４３が別の比較式で表され、ループの１回目の実行で「偽」になるような場合には、１サイクル目で、ポート３８に「０」が入力される。

【０１０２】

この場合、Ｖ１、Ｖ２およびＶ３は図４のように変化し、３サイクル目で節点４９において、ループの実行が終了される。図４では、Ｖ１、Ｖ２およびＶ３は、それぞれ、１サイクルの間だけ１となり、関数ｆ、ｇおよびｈの演算は、それぞれ１度ずつ実行されることになる。

【０１０３】

このようにして、本実施形態１のループ制御部３１を用いることによって、ループの繰り返し回数が小さい場合についても、正しく回路を動作させることができるようになる。

【０１０４】

次に、ループ処理部３０の節点４６、４７および４８について説明を行う。

【０１０５】

節点４６には、二つの値が入力されている。一つはポート４２からのｉの値であり、もう一つはポート３５からのＶ１の値である。節点４６は、Ｖ１の値が「１」のときにのみ、ｆ（ｉ）の処理を実行するようになっている。

【０１０６】

ここで、ｆが加算や乗算など、その結果が外部に対して作用しない処理の場合には、Ｖ１の値が「０」のときにｆを実行したとしても、その演算結果が使用されないだけであり、実害は生じない。したがって、このような場合には、Ｖ１によって節点４６の実行を制御する必要はなく、図１７の節点２５の場合と同様に、常にｆの処理を実行する節点としてもよい。

【０１０７】

一方、ｆがメモリ書き込みや通信など、外部に対して作用する処理の場合には、Ｖ１が「１」のときのみ、その処理が実行されるようにする必要がある。Ｖ１が「０」のときにｆの処理が実行されると、外部からみたときにループの振舞いが変わってしまうからである。

【０１０８】

具体的には、メモリ書き込みの場合にはイネーブル信号に対してＶ１との論理積を求め、通信の場合には通信要求信号に対してＶ１との論理積を求めるというような、簡単な回路を追加することによって対応することができる。以上のことは節点４７および４８についても、節点４６の場合と同様である。

【０１０９】

次に、節点５０および５１について説明を行う。

【０１１０】

この節点５０および５１は、ｆ（ｉ）の処理に並行して、ｇ（ｉ－１）およびｈ（ｉ－２）の処理を行うために設けられている。例えば、ｆ（３）と同時にｇ（２）およびｈ（１）が実行される。ｉが「１」のときには、ループ制御変数Ｖ２とＶ３が「０」であるため、ｆ（１）は実行されるが、同時にｇ（０）およびｈ（－１）が実行されることはない。

【０１１１】

なお、上述したように、ｇおよびｈがループ外に対して作用しない処理である場合には、ｇ（０）およびｈ（－１）が実行されても何ら実害は生じないため、ループ制御変数Ｖ２およびＶ３によって制御する必要もない。

【０１１２】

また、ｉが１２の場合には、Ｖ１およびＶ２が「０」であるため、ｈ（１０）

のみが実行され、ｆ（１２）およびｇ（１１）は実行されない。同様に、ｆおよびｇが外部に対して作用しない処理であれば、ｆ（１２）およびｇ（１１）を実行してもよい。

【０１１３】

（実施形態２）

本実施形態２では、図５の動作記述に対して、本発明のループ処理のパイプライン化方法を適用した例について説明する。

【０１１４】

図５はループ処理を含む動作記述の他の例を示す図である。図５において、この動作記述は、ｉ＝０からｉを一づつインクリメントしながらｉ＜１０の条件を満たす間、配列ａに対してｉを書き込み、ｉとｊを加算した結果を配列ｂに対して書き込むという処理を表している。ｊの初期値は５であり、次の繰り返しではｉ＋ｊがｊとして用いられる。

【０１１５】

図６は、図５の動作記述について、ループ処理をパイプライン化していない場合のＣＤＦＧを示す図である。

【０１１６】

図６において、矩形６０はループ（ループ処理部６０）を表し、ループ処理部６０内のＣＤＦＧが繰り返し実行される。矩形６０の上端に示すポート６３および６４はそれぞれ変数ｉおよびｊの値を表している。ポート６３に格納された変数ｉの値は、インクリメント演算節点６７でインクリメントされて＋１加えられた値となり、枝７１を介してポート７３に至る。ポート７３に至ったデータは、次の繰り返しで再びポート６３に戻されて変数ｉとして使用される。

【０１１７】

節点６５では終了条件の比較が行われ、真（ｉ≦１０）であれば「１」、偽であれば「０」が出力される。また、節点６６では、入力が「０」の場合に、図示しないコントローラに対してループ処理部６０の処理を終了し、次の状態に移行するように指示が与えられる。

【０１１８】

節点６１および６２は定数を出力する節点であり、ループ変数ｉおよびｊの初期値「０」および「５」が与えられる。これらの値はループ処理部６０の１回目の繰り返しで用いられ、２回目の繰り返し以降は、ポート７３および７４の値が用いられる。

【０１１９】

節点６８は、配列ａに対する書き込みを表す。この節点６８には、二つの入力が設けられており、左側の入力は配列のインデックス、右側の入力は書き込まれるデータを表す。ここでは、変数ｉが配列ａに書き込まれる。また、節点６９は、加算を表し、二つの入力ｉおよびｊの値が加算される。その加算結果は、枝７２を介してポート７４に至り、次の繰り返しで再びポート６４に戻されて変数ｊとして使用される。節点７０は、配列ｂに対する書き込みを表し、左側の入力は配列のインデックス、右側の入力は書き込まれるデータを表す。ここでは、節点６９からの加算結果が配列ｂに書き込まれる。

【０１２０】

図６のＣＤＦＧにおいて、節点６７および６８はステップ１に、節点６９はステップ２に、節点７０はステップ３に、それぞれスケジューリングされている。

【０１２１】

このスケジューリングの場合、１回の繰り返しのために３サイクルを要し、ループを１０回繰り返すには３０サイクル必要である。したがって、ループを１０回繰り返すためには、３０サイクルが必要となる。

【０１２２】

ところで、図１４のような単純な動作記述の場合には、図１に示すようなパイプライン化されたＣＤＦＧを作成することは比較的容易であるが、図５に示す動作記述のように、各ステップ間でデータの受け渡しがあるような場合には、次の繰り返しへのデータ受け渡しも考慮して、パイプライン化されたときのデータの受け渡しを決定する必要があるため、複雑である。

【０１２３】

このような場合には、図６に示すパイプライン化していない場合のループ処理のＣＤＦＧを元に、パイプライン化されたループのＣＤＦＧを作成する方法が有

効である。この方法について、図７を用いて説明する。

【０１２４】

図７は、本実施形態２のＣＤＦＧのパイプライン化処理手順を示すフローチャートである。

【０１２５】

コンピュータ（計算）上のデータベース１１４に記憶されたパイプライン化されていないループ処理を含むＣＤＦＧに対して、図７に示すような方法をプログラム化して適用することによって、パイプライン化されたループ処理のＣＤＦＧをコンピュータ（計算機）上（例えばＣＤＦＧ生成手段１１１Ａ）で自動的に生成することが可能となる。

【０１２６】

以下に、図７に示すループ処理のパイプライン化方法の各ステップについて説明する。この例でも、パイプライン段数は「３」としている。

【０１２７】

まず、ステップＳ１１の前処理では、図６に示すパイプライン化されていないＣＤＦＧに対して、図８Ａに示すように、ステージの境界部分とグラフの枝が交差する箇所に新たにポート８１、８２、８３および８４を追加する。パイプライン段数は３であるため、図６のステップ１～３は、それぞれ、図８Ａのステージ１～３に対応する。

【０１２８】

ここで、図６に示すループ処理の繰り返し用ポート７３および７４に接続されている枝７１および７２は、ループの次の繰り返しへのデータ転送のために用いられる枝である。

【０１２９】

次のステップＳ１２では、同一繰り返し内でのデータ転送についてのみ考慮されるため、枝７１および７２は、ここでは一旦削除される。この枝７１および７２については、ステップＳ１３で考慮されるようになっている。

【０１３０】

次に、ステップＳ１２のデータフローグラフの変形処理では、各ステージの処

理が並列に動作するように、ステップＳ１１で前処理された各ステージ内のＣＤＦＧを、図８Ｂに示すように、横に並べる。図８Ｂの例では、ステージ１からステージ３が左から右へと並べられている。

【０１３１】

図８Ｂにおいて、ポート６３および６４はそれぞれ、変数ｉおよびｊに対応する。パイプライン化前の図８Ａにおいて、枝８５はポート６３と８１とを接続しており、ポート８３のデータは、次のサイクルでポート８１から使用される。

【０１３２】

これに対して、パイプライン化された図８Ｂでは、ポート６３の値は、次のループの繰り返しでポート８１に転送される必要がある。そのため、図８Ｂにおいては、枝８５ａによって、ポート６３が、ポート８１ではなく、ポート８１ａに接続されている。これにより、ポート６３の変数ｉの値は、次の繰り返しでポート８１ａからポート８１に転送され、節点６９に入力されることになる。

【０１３３】

その他の枝についても同様に、次のループの繰り返しでデータが転送されるようにポートが接続される。例えば、図８Ａのポート６４と８２とを接続する枝８６は、図８Ｂではポート６４と８２ａとを接続する枝８６ａとされている。また、図８Ａでポート８１と８３とを接続する枝８７は、図８Ｂではポート８１とポート８３ａとを接続する枝８７ａとされ、図８Ａで節点６９とポート８４とを接続する枝８８は、図８Ｂでは節点６９とポート８４ａとを接続する枝８８ａとされている。

【０１３４】

次に、ステップＳ１３の繰り返し間データ転送枝の接続処理では、図６に示すループ処理の繰り返し用ポートに接続されている枝７１および７２に相当する枝７１ａおよび７２ａを、図８Ｃに示すように接続する。

【０１３５】

この場合、図６に示すパイプライン化されないＣＤＦＧにおいて、枝に対してデータを出力する節点と、そのデータを次の繰り返しで使用する節点のステージの差を考慮する。

【０１３６】

図９は、二つの節点のステージの差に応じて、どのようにパイプライン化すればよいかを示す図である。

【０１３７】

図９（ａ）では、ループがパイプライン化されないときに、データが出力されるＢ節点１０２とデータが入力されるＡ節点１０１とが同一ステージに含まれている場合を示している。このＣＤＦＧがパイプライン化されると、１ステージが１サイクルで実行される場合、各ステージは毎サイクル実行されることになるため、Ｂ節点１０２の処理が実行されて生成されたデータは、１クロックサイクル後に、ループの次の繰り返しのＡ節点１０１で使われることになる。これによって、図９（ｂ）に示すパイプライン化されたＣＤＦＧでは、Ｂ節点１０２から出力されたデータが、次のループの繰り返しでＡ節点に入力されて利用されるように、Ｂ節点１０２をポート１０４に接続し、次のループの繰り返しでポート１０４からのデータが供給されるポート１０３にＡ節点１０１を接続する。

【０１３８】

一方、図９（ｃ）は、ループがパイプライン化されないときに、データが出力されるＢ節点１０２がデータが入力されるＡ節点１０１よりも１ステージ前にスケジューリングされている場合を示している。このＣＤＦＧがパイプライン化されると、Ｂ節点１０２の処理が実行されて生成されたデータは、２クロックサイクル後に、ループの次の繰り返しのＡ節点１０１で使われることになる。これによって、図９（ｄ）に示すパイプライン化されたＣＤＦＧでは、Ｂ節点１０２から出力されたデータが、ループを２回繰り返した後にＡ節点１０１に入力されて利用されるように、Ｂ節点１０２を次のステップのポート１０７に接続し、ポート１０７からのデータが次のループの繰り返しで入力されるポート１０５をポート１０８に接続し、ポート１０８からのデータが次のループの繰り返しで入力されるポート１０６にＡ節点１０１を接続する。

【０１３９】

これと同様に、節点Ｂが節点Ａのｎステージ前にスケジューリングされている場合には、ループをｎ＋１回繰り返した後でデータが使用されるように、図９（

ｄ）に示すポート１０５および１０８のような繰り返しを追加する。

【０１４０】

さらに、図９（ｅ）は、ループがパイプライン化されないときに、データが出力されるＢ節点１０２がデータが入力されるＡ節点１０１よりも１ステージ後にスケジューリングされている場合を示している。このＣＤＦＧがパイプライン化されると、Ｂ節点１０２と次のループの繰り返しの節点Ａ１０１とは、同一のクロックサイクルで実行される。これによって、図９（ｆ）に示すパイプライン化されたＣＤＦＧでは、Ｂ節点１０２とＡ節点１０１とが直接接続される。

【０１４１】

なお、パイプライン化されないときに、データが出力するＢ節点１０２がデータが入力されるＡ節点１０１よりも２ステージ以上後にスケジューリングされている場合には、ループをパイプライン化すること自体が不可能であるため、ここでは考えないこととする。

【０１４２】

例えば図６に示す枝７１の場合、その枝に対してデータを出力する節点は節点６７であり、そのデータを次の繰り返して使用する節点も節点６７であるため、実行されるステージの差は「０」である。したがって、図９（ｂ）に示すように枝を接続すればよい。ここでは、図８Ｃに示すように、ポート６３、節点６７および８３ａが枝７１ａによって接続されている。

【０１４３】

また、枝７２の場合にも、その枝に対してデータを出力する節点は節点６９であり、そのデータを次の繰り返しで使用する節点も節点７０であるため、ステージの差は「０」である。したがって、図９（ｂ）に示すように枝を接続すれば良いが、この場合には、接続先のポート８２ａに別の枝８６ａが接続されている。このため、図８Ｃに示すように、セレクタ節点８９を挿入して枝７２ａを接続する。このセレクタ節点８９は、１サイクル目のみ、ポート６４の値を出力し、それ以外は節点６９の演算結果をポート８２ａに出力するようにコントロールされる。ポート６４には、図５に示す動作記述の変数ｊの初期値が入力されており、最初のループの繰り返しではｊの初期値が使用され、それ以降はループ内部で演

算された値である節点６９からの出力が使用される。

【０１４４】

次に、図７のステップＳ１４のループ制御部の追加処理では、図８Ｄに示すように、ＣＤＦＧにループ制御部９０が加えられる。

【０１４５】

図８Ｄにおいて、図６のＣＤＦＧが３段にパイプライン化されるため、三つのループ制御変数Ｖ１、Ｖ２およびＶ３が必要であり、ポート９１、９２および９３から、Ｖ１、Ｖ２およびＶ３がそれぞれ出力される。

【０１４６】

制御変数Ｖ１は、ループがパイプライン化されたときにステージ１に含まれる処理を制御する制御信号であり、制御変数Ｖ２は、ステージ２に含まれる処理を制御する制御信号であり、制御変数Ｖ３は、ステージ３に含まれる処理を制御する制御信号である。制御変数Ｖ１は、初期値として１が与えられ、制御変数Ｖ２およびＶ３は、初期値として０が与えられる。

【０１４７】

制御変数Ｖ１およびＶ２の値は、次の繰り返しでそれぞれＶ２およびＶ３に与えられるように、ポート９１とポート９２ａとが枝９４ｂを介して接続され、ポート９２とポート９３ａとが枝９５ｂを介して接続されている。

【０１４８】

また、ポート９１ａには、枝９４ｃを介して、比較節点６５によるループ終了条件の比較結果が入力されており、ループ終了条件が「真」になったときに、ポート９１ａに「０」が入力される。

【０１４９】

さらに、終了を判定する節点９７は、枝９５ｃおよび９６ｂを介して、ポート９２およびポート９３に接続されており、制御変数Ｖ２が「０」、かつ、Ｖ３が「１」のときにループの処理を終了するように、図示しないコントローラに指示が与えられる。

【０１５０】

ポート９１および９３からの制御変数Ｖ１およびＶ３は、それぞれ、枝９４ａ

および９６ａを介してステージ１の節点６８およびステージ３の節点７０に接続されている。制御変数Ｖ１の値は、図３に示すように、１サイクル目～１０サイクル目が「１」であり、１１サイクル目および１２サイクル目は「０」である。したがって、１サイクル目～１０サイクル目では、節点６８において配列ａに対する書き込み処理が行われ、１１サイクル目および１２サイクル目では、節点６８の処理は行われない。また、制御変数Ｖ３の値は、図３に示すように、１サイクル目および２サイクル目が「０」であり、３サイクル目～１２サイクル目は「１」である。したがって、１サイクル目および２サイクル目では節点７０の処理は行われず、３サイクル目～１２サイクル目では、節点７０において配列ｂに対する書き込み処理が行われる。このように、配列ａに書き込みが行なわれる節点６８および配列ｂに書き込みが行なわれる節点７０に対して、プロローグ部およびエピローグ部の動作を制御することができる。

【０１５１】

また、ポート９２からの制御変数Ｖ２は、枝９５ａを介して、セレクタ節点８９に入力されている。制御変数Ｖ２の値は、図３に示すように、１サイクル目が「０」で、２サイクル目から１１サイクル目は「１」である。したがって、１サイクル目は、セレクタ節点８９において左の入力に接続されるポート６４の値が出力され、２サイクル目以降は右の入力に接続される節点６９の演算結果が出力される。このように、１サイクル目のみポート６４に接続される外部からの初期値が選択され、それ以降は、ループ内部で演算が行われた結果が選択される。制御変数Ｖ２は、１２サイクル目が「０」であるため、１２サイクル目はポート６４の値が選択されるが、１２サイクルでループの処理が終了されるため、１２サイクル目のセレクタ節点８９の動作は、回路全体の動作に対して無関係である。

【０１５２】

即ち、ＣＤＦＧ生成手段１１１Ａの他の実施形態として、パイプライン化されていないＣＤＦＧに対して、ステージ境界部分とグラフの枝が交差する箇所に新たにポートを追加するポート追加手段と、並列処理を示すために各ステージを横方向に配置する横方向配置手段と、横方向に配置されたＣＤＦＧに対して、繰り返し間データ転送枝を接続処理する枝接続手段と、ループ処理部にループ制御部

を追加処理するループ制御部追加手段とを有している。

【０１５３】

以上のような処理手順により、パイプライン化されていないループのＣＤＦＧから、ループ処理がパイプライン化されたＣＤＦＧを得ることができる。

【０１５４】

このように、ループ処理がパイプライン化されたＣＤＦＧを生成し、図１０に示すように、アロケーション処理、データパス生成処理、コントローラ生成処理などを行うことによって、動作記述を実現するためのハードウェアを合成することが可能である。

【０１５５】

したがって、上記実施形態１，２によれば、ループ処理をパイプライン化するときに、ループ処理部３０または６０を構成する各パイプライン化ステージに対して、そのパイプライン化ステージに含まれる各節点の処理が実行されるクロックサイクルと、非ループ処理が実行されるクロックサイクルとを制御する制御信号Ｖ１～Ｖ３を各節点４６～４８または６８～７０に出力するループ制御部３１または９０を設けたＣＤＦＧを作成する。これによって、動作記述からハードウェア（設計回路図）を合成する動作合成において、少ない面積の増加でＣＤＦＧ中のループ処理をパイプライン化することができる。

【０１５６】

【発明の効果】

以上により、本発明によれば、ループ処理および非ループ処理が含まれている動作記述から、パイプライン化されたＣＤＦＧを生成する際に、ループ処理に含まれる各節点を各パイプライン化ステージ毎に分けて複数回のループ処理の１回毎に各パイプライン化ステージの処理を並行して実行することを示すコントロールデータフローグラフのループ処理部に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる制御信号を各節点に出力するループ制御部を設けることによって、従来のプロローグ部とエピローグ部の非ループ処理を設けなくても、この非ループ処理を、パイプライン化したループ処理部に実行させることができ、従来のループ処理のパイプライン化方法で問題となっているハード

ウェア構成の面積増大やコスト増加を防ぐことができる。また、プロローグ部の処理が終了する前にループ処理が終了されてしまう場合や、ループ処理の繰り返し回数が一定でない場合であっても、誤動作を招くことなく、ループ処理のパイプライン化を行うことができる。

【図面の簡単な説明】

【図１】

本発明の動作合成システムの実施形態における要部構成を示すブロック図である。

【図２】

図１４の動作記述に対して、本実施形態１のループ処理のパイプライン化方法によりループ処理をパイプライン化したＣＤＦＧを示す図である。

【図３】

図２のループ制御部の出力信号の一例を示す図である。

【図４】

図２のループ制御部の出力信号の他の一例を示す図である。

【図５】

図１４の動作記述とは別の動作記述の一例を示す図である。

【図６】

パイプライン化されないＣＤＦＧの一例を示す図である。

【図７】

図６のＣＤＦＧをパイプライン化する本実施形態２のパイプライン化処理手順を示すフローチャートである。

【図８Ａ】

ＣＤＦＧのパイプライン化処理手順（その１）を示す図である。

【図８Ｂ】

ＣＤＦＧのパイプライン化処理手順（その２）を示す図である。

【図８Ｃ】

ＣＤＦＧのパイプライン化処理手順（その３）を示す図である。

【図８Ｄ】

ＣＤＦＧのパイプライン化処理手順（その４）を示す図である。

【図９】

（ａ）および（ｂ）～（ｅ）および（ｆ）は、繰り返し間枝の接続処理例を説明するための図である。

【図１０】

従来の動作合成方法の処理手順を示すフローチャートである。

【図１１】

従来の動作記述の式の一例を示す図である。

【図１２】

図１１の動作記述に対応したＣＤＦＧを表す図である。

【図１３】

図１２のＣＤＦＧのデータ構造を説明するための図である。

【図１４】

動作記述の一例を示す図である。

【図１５】

従来のパイプライン化されないＣＤＦＧの一例を示す図である。

【図１６】

従来のパイプライン動作について説明するための図である。

【図１７】

従来手法によりパイプライン化されたＣＤＦＧの一例を示す図である。

【符号の説明】

３０、６０　ループ処理部

３２、３３、３４、４１、６１、６２　定数出力節点

４２、４５、６３、６４、６３ａ、６４ａ、７３、７４、８１、８２、８１ａ、８２ａ、８３、８４、８３ａ、８４ａ　ループ処理部のポート

４３、６５　終了条件比較節点

４４、６７　インクリメント演算節点

４９、６６　終了判定節点

４６～４８　関数演算節点

２０　プロローグ部

５０、５１　減算節点

２９　エピローグ部

３１、９０　ループ制御部

３５～４０、９１～９３、９１ａ～９３ａ　ループ制御部のポート

６８、７０　配列書き込み節点

６９　加算節点

７１、７２、７１ａ、７２ａ、８５～８８、８５ａ～８８ａ　枝

８１～８４　ステージの境界部分とＣＤＦＧの枝が交差する箇所のポート

８９　セレクタ節点

９４ａ、９４ｂ、９４ｃ、９５ａ、９５ｂ、９５ｃ、９６ａ、９６ｂ　ループ制御部に接続された枝

１００　動作合成システム

１０１、１０２　節点

１０３～１０８　ポート

１１１　制御部

１１１Ａ　ＣＤＦＧ生成手段

１１１Ｂ　スケジューリング手段

１１１Ｃ　アロケーション手段

１１１Ｄ　データパス生成手段

１１１Ｅ　コントローラ生成手段

１１２　ＲＯＭ

１１３　ＲＡＭ

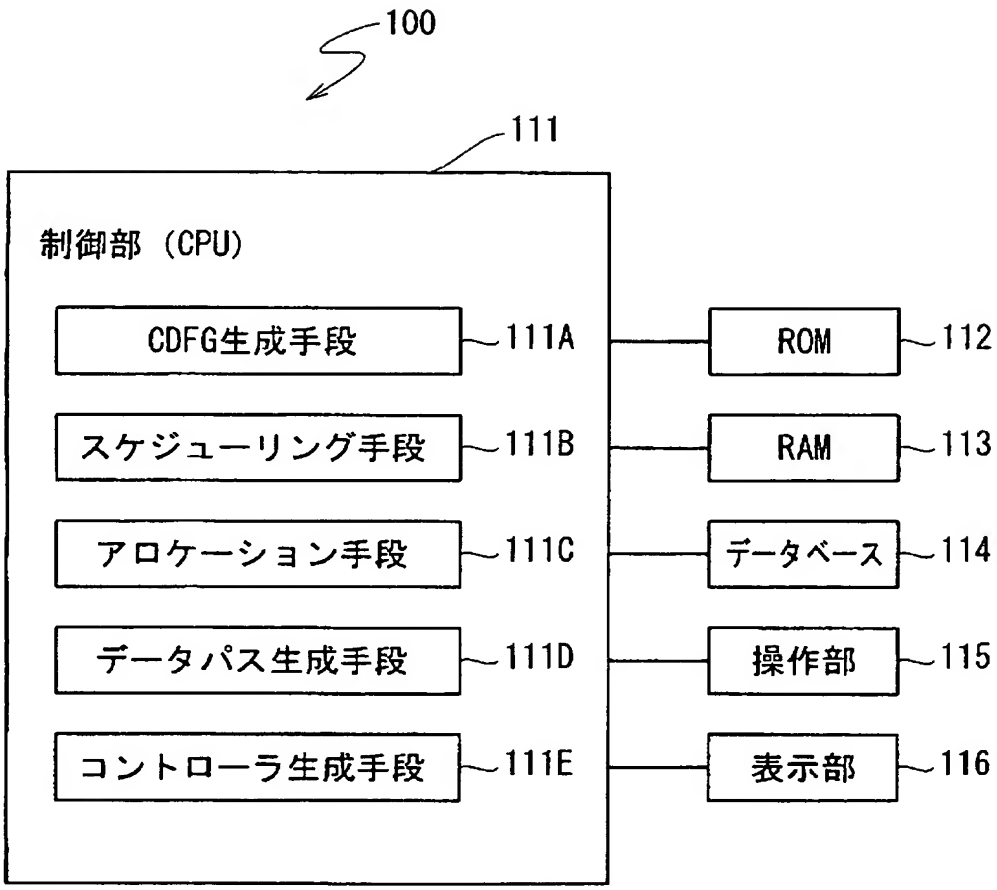
１１４　データベース

１１５　操作部

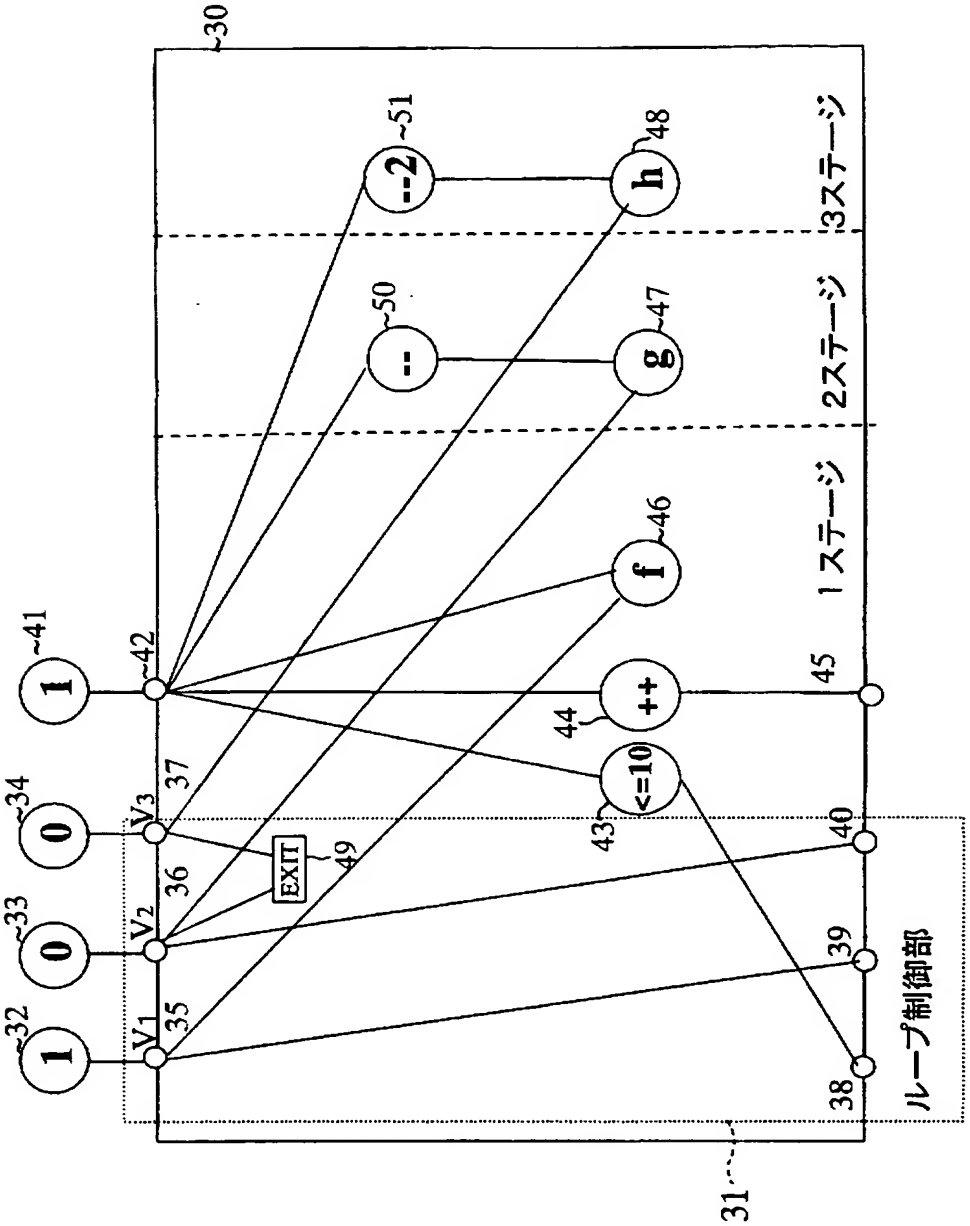
１１６　表示部

【書類名】　図面

【図１】


100
111
制御部（CPU）
CDFG生成手段
111A
スケジューリング手段
111B
アロケーション手段
111C
データパス生成手段
111D
コントローラ生成手段
111E
ROM
112
RAM
113
データベース
114
操作部
115
表示部
116

【図２】


30
-2
51
h
48
3ステージ
-
50
g
47
2ステージ
f
46
1ステージ
1
41
42
++
44
45
<=10
43
0
34
V3
37
EXIT
49
36
0
33
V2
40
39
1
32
V1
35
38
ループ制御部
31

【図３】

サイクル	V1	V2	V3
1	1	0	0
2	1	1	0
3～10	1	1	1
11	0	1	1
12	0	0	1

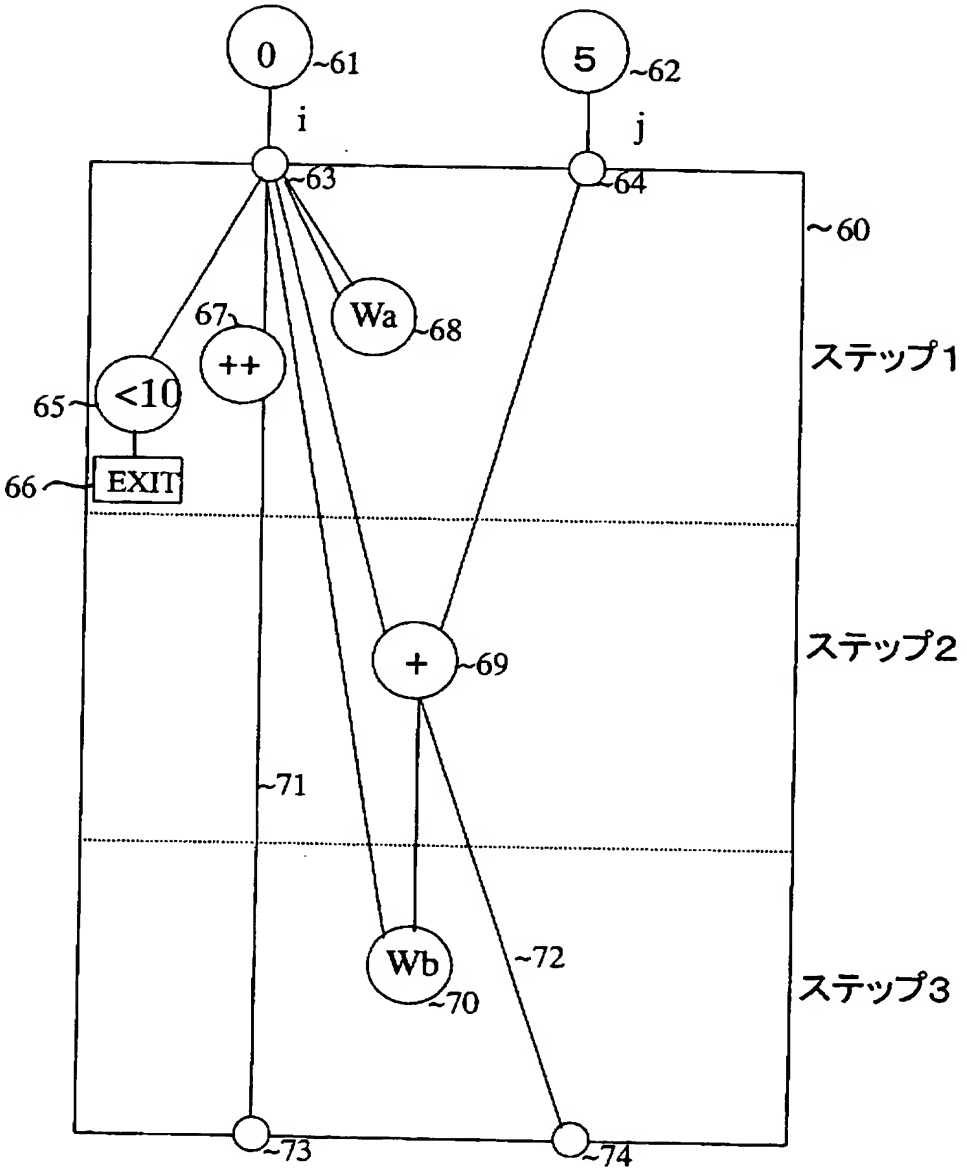
【図４】

サイクル	V1	V2	V3
1	1	0	0
2	0	1	0
3	0	0	1

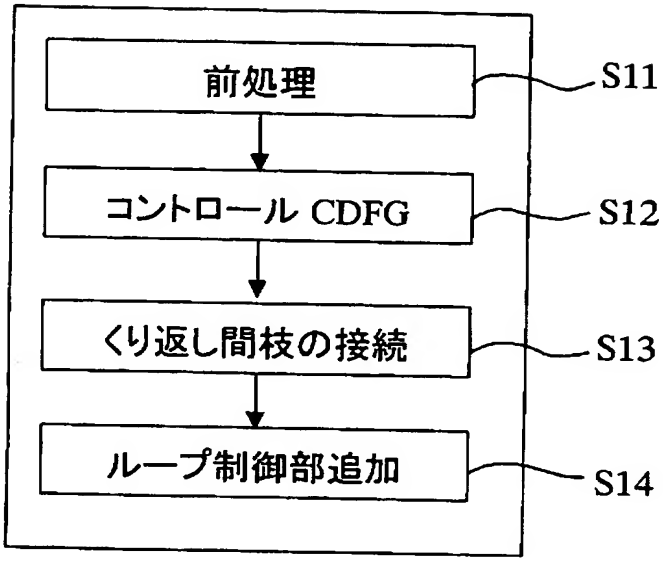
【図５】

```
j = 5;
for (i = 0; i<10 ; i++)
{
   a[i] = i ;
   j += i ;
   b[i] = j ;
}
```

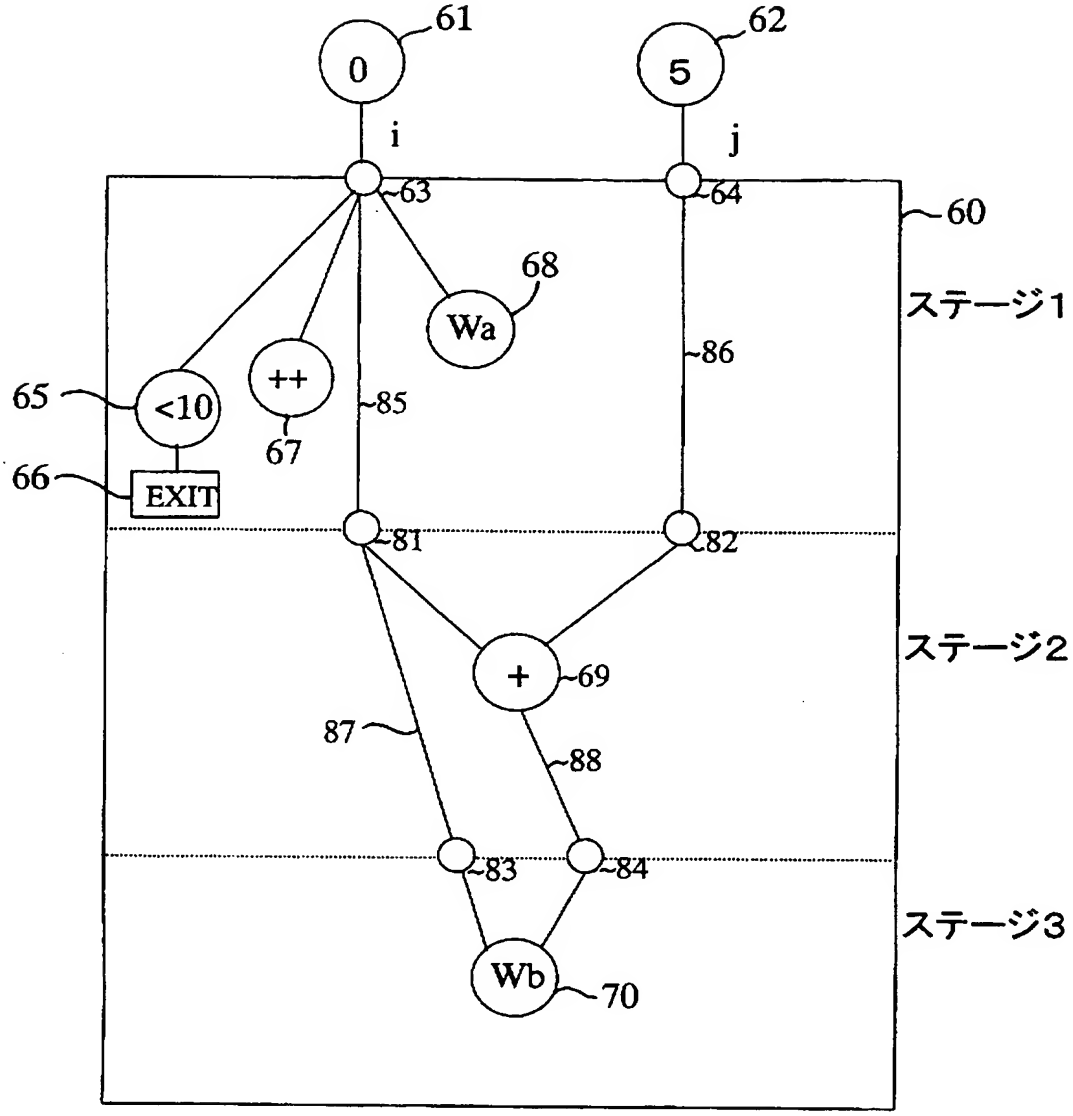
【図６】


0
~61
i
5
~62
j
~63
~64
~60
67
++
Wa
~68
ステップ1
65
<10
66
EXIT
+
~69
ステップ2
~71
Wb
~70
~72
ステップ3
~73
~74

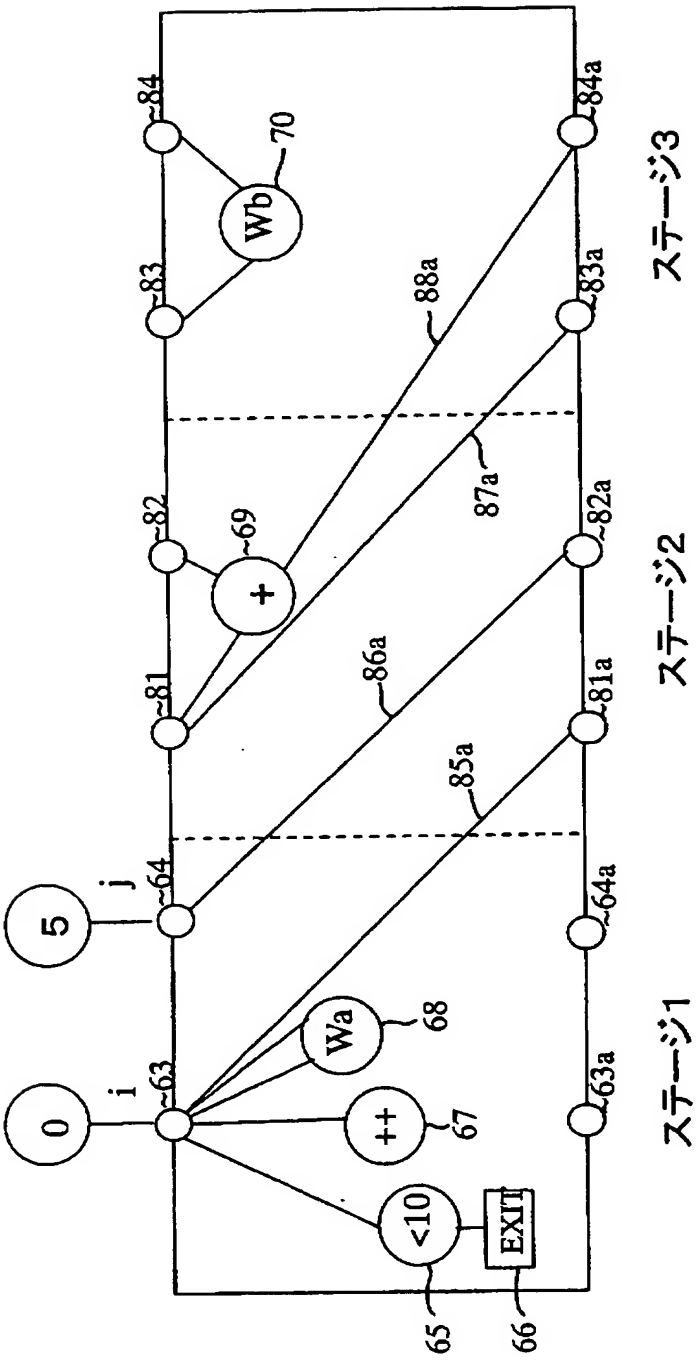
【図７】



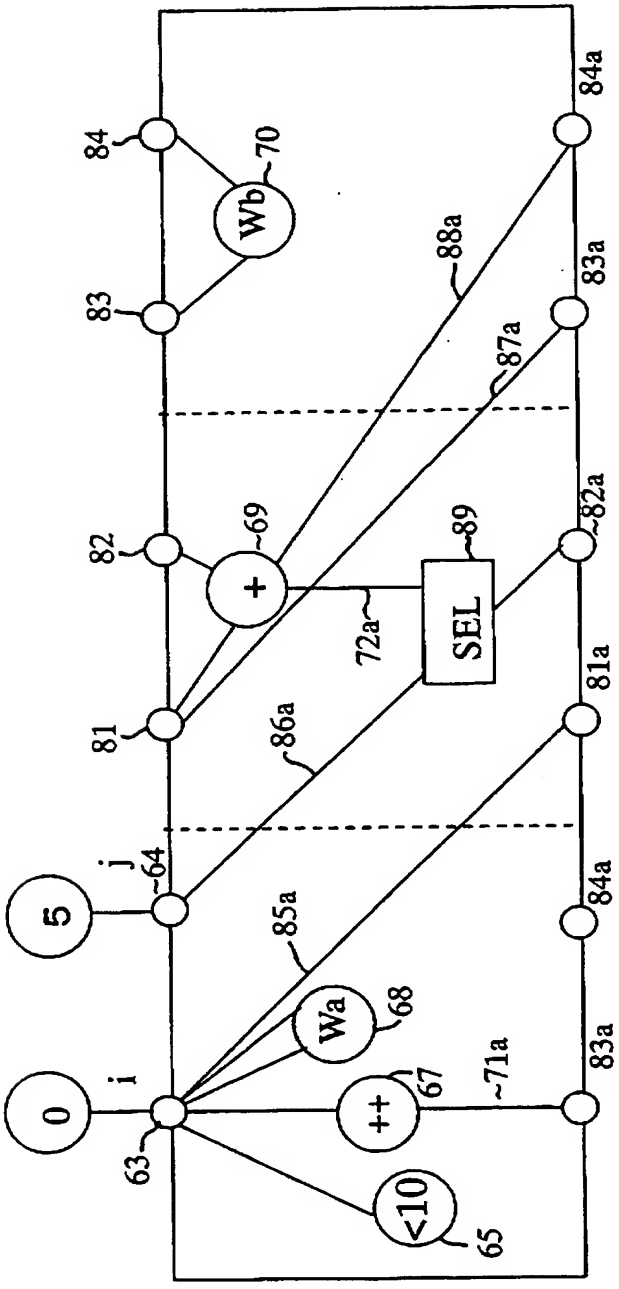
【図８Ａ】


61
0
62
5
i
j
63
64
60
ステージ1
68
Wa
++
67
65
<10
66
EXIT
85
86
81
82
ステージ2
+
69
87
88
83
84
ステージ3
Wb
70

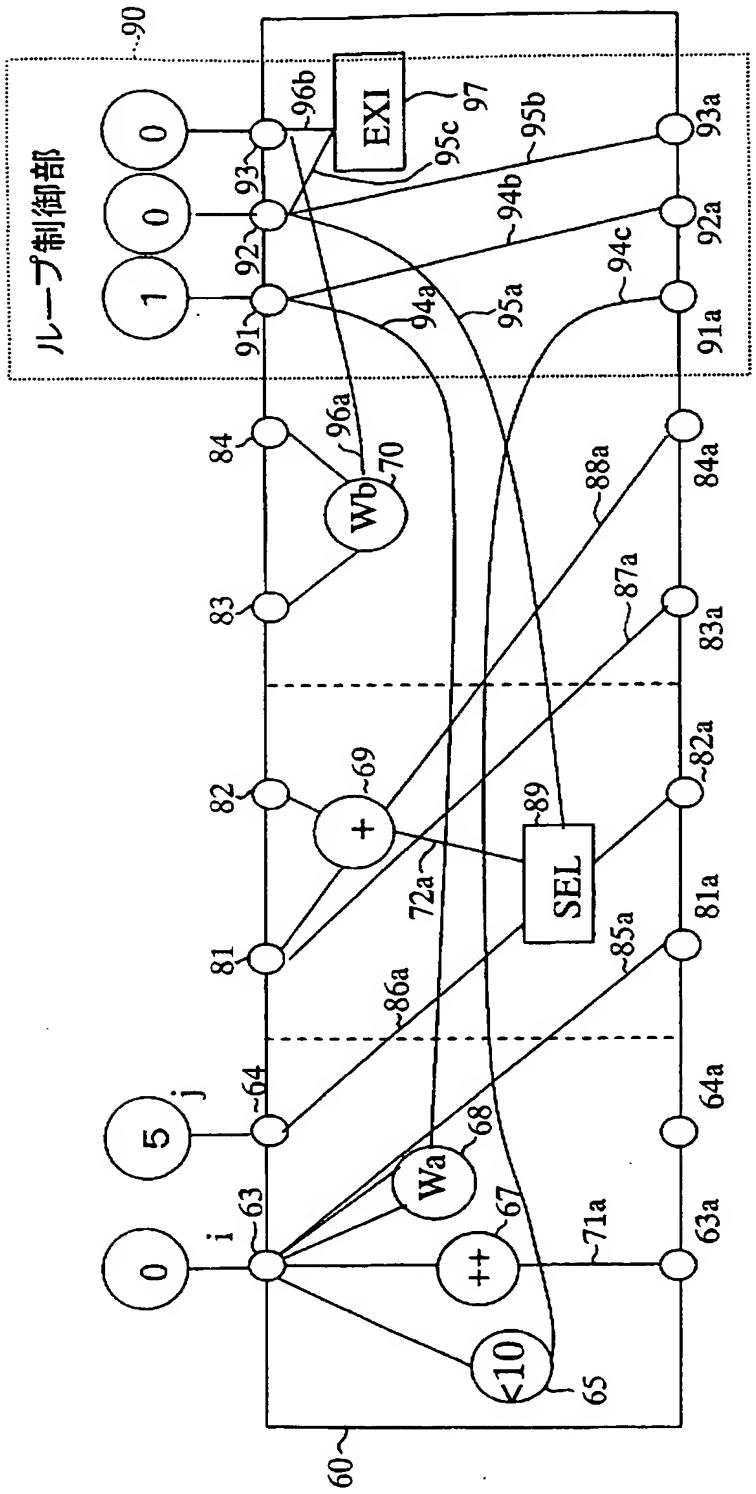
【図８Ｂ】


ステージ3
ステージ2
ステージ1
84
84a
70
Wb
83
83a
88a
87a
82
82a
69
+
81
81a
86a
85a
64
64a
j
5
Wa
68
i
63
63a
0
++
67
<10
EXIT
65
66

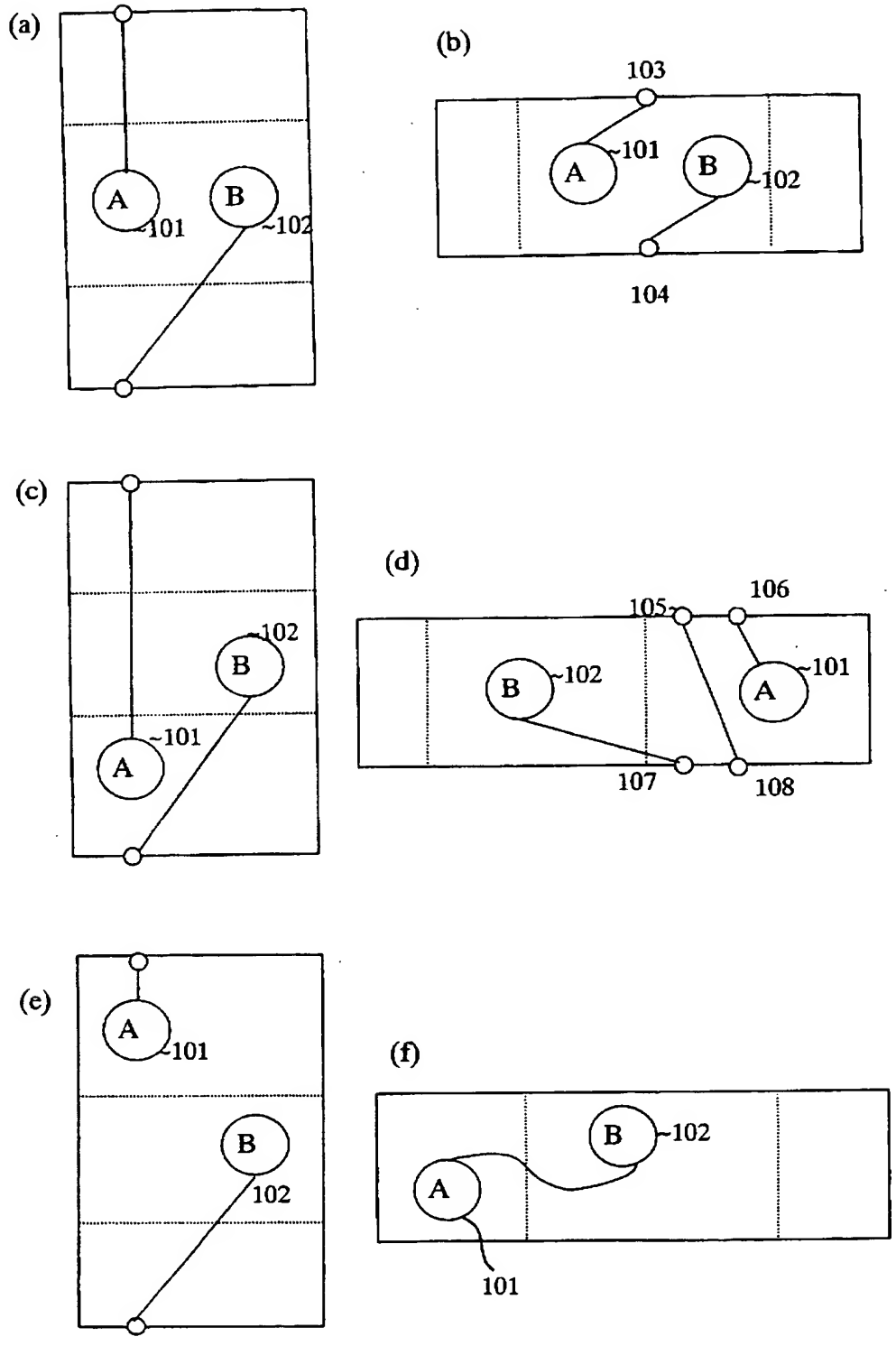
【図８Ｃ】


84
70
Wb
83
88a
84a
83a
87a
82
69
+
89
SEL
82a
72a
81
81a
86a
j
64
5
85a
84a
Wa
68
i
0
67
++
71a
83a
63
<10
65

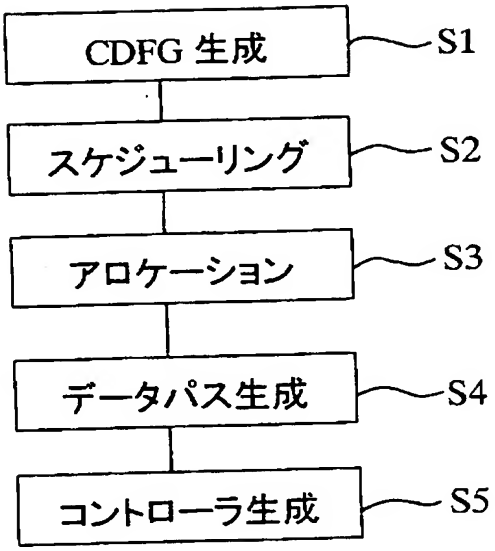
【図８Ｄ】


90
ループ制御部
0
0
1
93
92
91
96b
EXI
97
95c
95b
94b
94c
94a
95a
93a
92a
91a
84
96a
70
Wb
88a
84a
83
87a
83a
82
69
+
89
SEL
72a
82a
81
81a
85a
86a
64
j
5
i
0
63
Wa
68
67
++
71a
64a
63a
<10
65
60

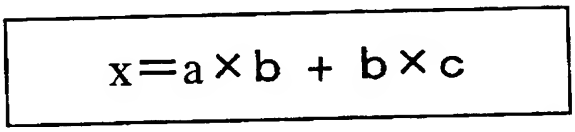
【図９】


(a)
A
101
B
102
(b)
103
A
101
B
102
104
(c)
102
B
101
A
(d)
106
105
B
102
A
101
107
108
(e)
A
101
B
102
(f)
B
102
A
101

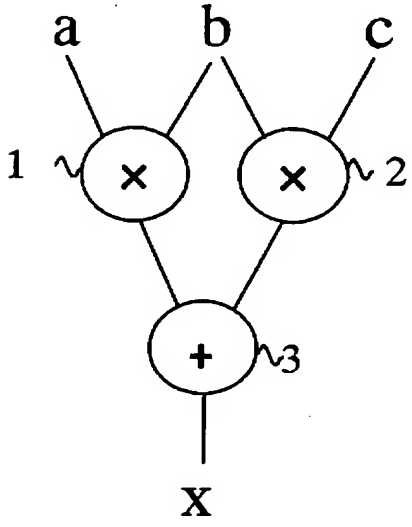
【図１０】


CDFG 生成
S1
スケジューリング
S2
アロケーション
S3
データパス生成
S4
コントローラ生成
S5

【図１１】


x＝a×b ＋ b×c

【図１２】


a
b
c
1
×
×
2
+
3
x

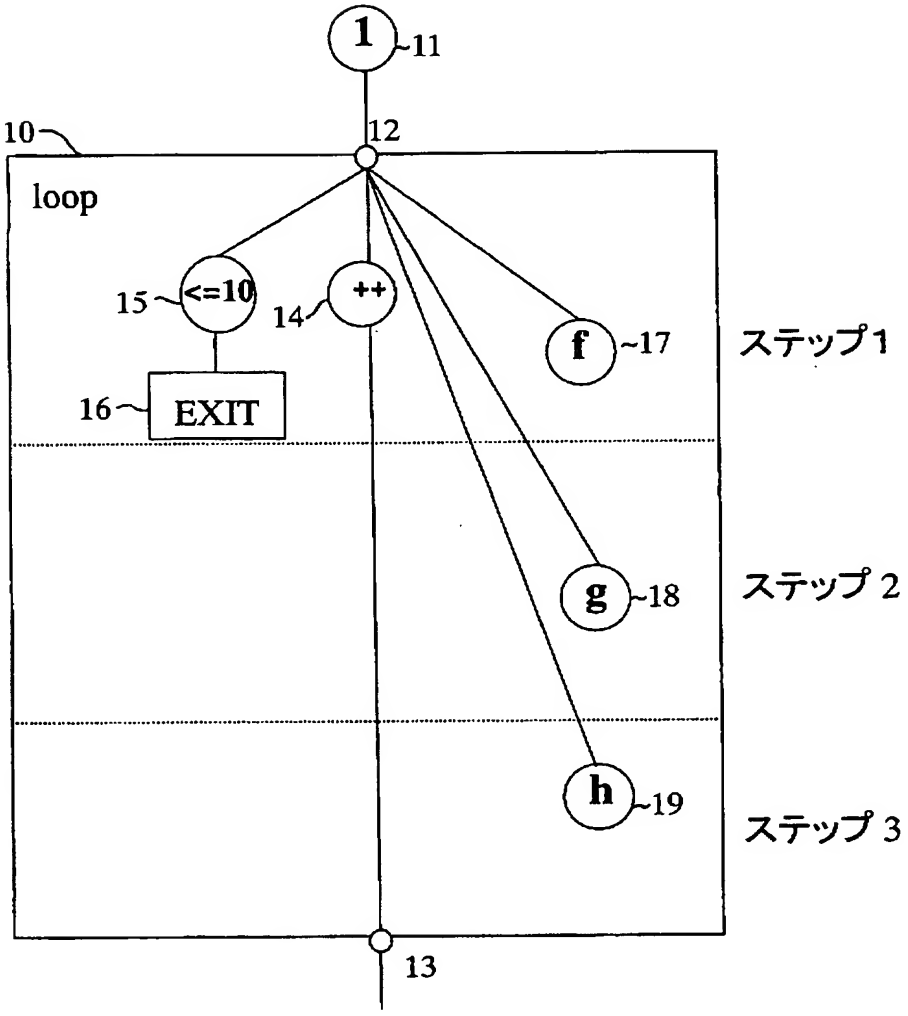
【図１３】

```
struct Node
  {
   int   node_id;
   int   in_edge[2];
   int   out_edge[1];
   int   op_type;
   }
struct Edge
  {
   int   edge_id;
   int   from_node;
   int   to_node;
   }
```

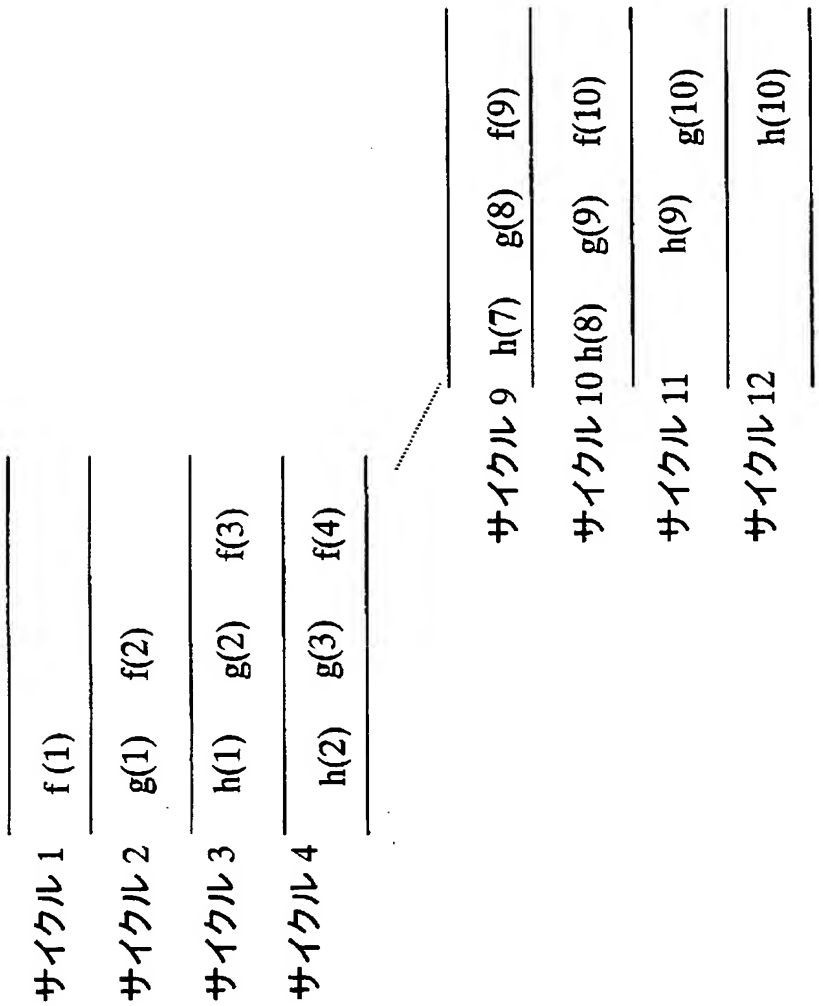
【図１４】

```
for( i =0; i <=10 ; i++)
{
    f(i);
    g(i);
    h(i);
}
```

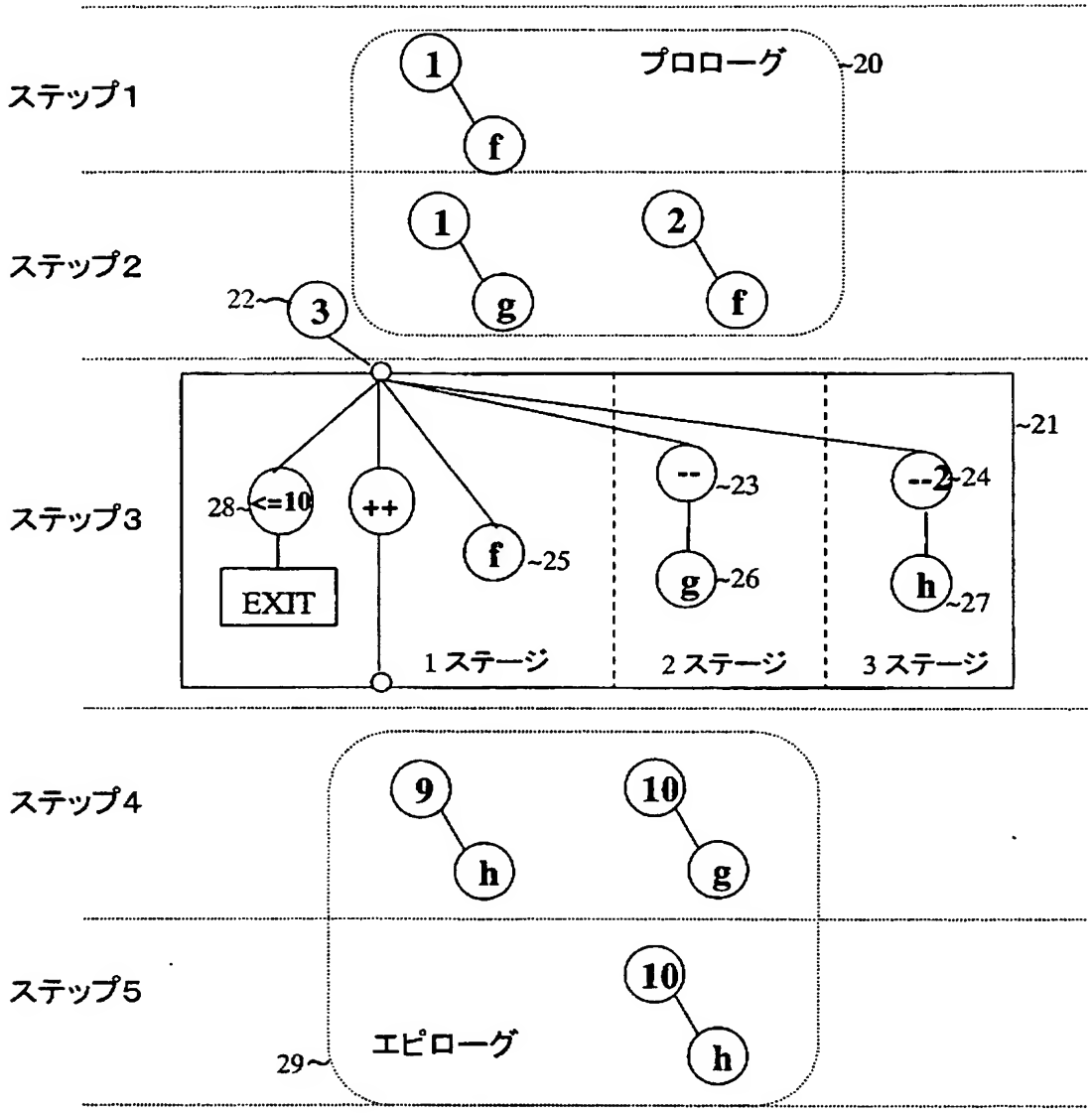
【図１５】



【図１６】


サイクル１　f(1)
サイクル２　g(1)　f(2)
サイクル３　h(1)　g(2)　f(3)
サイクル４　h(2)　g(3)　f(4)
サイクル９　h(7)　g(8)　f(9)
サイクル１０　h(8)　g(9)　f(10)
サイクル１１　h(9)　g(10)
サイクル１２　h(10)

【図１７】


ステップ1
ステップ2
ステップ3
ステップ4
ステップ5
プロローグ
~20
1
f
1
g
2
f
22~
3
~21
28~
<=10
++
f
~25
--
~23
--
~24
g
~26
h
~27
EXIT
1ステージ
2ステージ
3ステージ
9
h
10
g
10
h
29~
エピローグ

【書類名】　要約書

【要約】

【課題】　動作記述からハードウェアを合成する動作合成において、少ない面積の増加でＣＤＦＧ中のループ処理をパイプライン化する。

【解決手段】　ループ処理をパイプライン化するときに、ループ処理に含まれる各節点を各パイプライン化ステージ毎に分けて複数回のループ処理の１回毎に各パイプライン化ステージの処理を並行して実行することを示すコントロールデータフローグラフのループ処理部３０に、ループ処理および非ループ処理のうち少なくとも非ループ処理を行わせる制御信号を各節点に出力するループ制御部３１を設けたＣＤＦＧを生成する。

【選択図】　図１

認定・付加情報

特許出願の番号	特願２００３－１２０６００
受付番号	５０３００６９０７９２
書類名	特許願
担当官	小野寺　光子　　１７２１
作成日	平成１５年　４月２５日

＜認定情報・付加情報＞

【特許出願人】
- 【識別番号】　000005049
- 【住所又は居所】　大阪府大阪市阿倍野区長池町２２番２２号
- 【氏名又は名称】　シャープ株式会社

【代理人】　申請人
- 【識別番号】　100078282
- 【住所又は居所】　大阪市中央区城見１丁目２番２７号　クリスタルタワー１５階
- 【氏名又は名称】　山本　秀策

【選任した代理人】
- 【識別番号】　100062409
- 【住所又は居所】　大阪府大阪市中央区城見１丁目２番２７号　クリスタルタワー１５階　山本秀策特許事務所
- 【氏名又は名称】　安村　高明

【選任した代理人】
- 【識別番号】　100107489
- 【住所又は居所】　大阪市中央区城見一丁目２番２７号　クリスタルタワー１５階　山本秀策特許事務所
- 【氏名又は名称】　大塩　竹志

次頁無

特願２００３－１２０６００

出　願　人　履　歴　情　報

識別番号　　［０００００５０４９］

１．変更年月日　　１９９０年　８月２９日
［変更理由］　　新規登録
住　所　　大阪府大阪市阿倍野区長池町２２番２２号
氏　名　　シャープ株式会社